INSTITUTO SUPERIOR DE PROFESORADO Nº 63 NATALIA QUESSÚS

TECNICATURA SUPERIOR EN DESARROLLO DE SOFTWARE

CUADERNILLO PROPEDÉUTICO INGRESANTES 2020

Contenido

BIENVENIDA!

Este material tiene como propósito que los ingresantes a la Carrera de **TECNICATURA SUPERIOR EN DESARROLLO DE SOFTWARE** dispongan del núcleo de conocimientos básicos, iniciarlos en el conocimiento de aspectos de la cultura institucional y presentar los desafíos del ser Técnico.

Por Un lado encontrarán un apartado de Formación General que contribuye a pensar en qué consiste ser estudiantes en el nivel y futuros profesionales técnicos.

Por el otro, con respecto a la Formación Específica podrán aproximarse a lo que han elegido para aprender un vistazo a los planes y contenidos del saber académico específico. También

La intención es compartir información y actividades que nos permitan descubrir y apreciar algunos conceptos, actitudes y valores que forman parte de la carrera que han elegido.

Este tiempo de propedéutico con permitirá a sus docentes conocerlos, adaptar nuestra propuesta a sus necesidades, y acompañarlos en el proceso de integración al instituto Superior

Este documento es un aporte para el conocimiento de los saberes que cada uno posee, como así también un espacio de reflexión para "Aprender juntos".

Lo importante es poder advertir cuáles son los saberes básicos que debemos construir, como así también los procedimientos necesarios para resolver situaciones de aprendizaje que te permitan avanzar en tu carrera

Conociendo el Instituto Superior de Profesorado Nº 63

El Instituto Superior de Profesorado Nº 63 "Natalia Quessús" se independizo en el año 2014 , del Instituto de Educación Superior Nº4 de Reconquista,

El Instituto cuenta con más de 600 estudiantes que cursan en 4 carreras que se dictan en 2 sedes:

- Profesorado de Nivel Primario ,
- Profesorado de Lengua y Literatura
- Tecnicatura Superior en Desarrollo de Software.
- Tecnicatura Superior en Mantenimiento industrial (Sede Avellaneda)

Para conocer más sobre su organización y carreras, te proponemos que consultes este link:

https://isp4lt-sfe.infd.edu.ar/

Además nos comunicamos con nuestros estudiantes a través de Facebook del instituto:

https://www.facebook.com/**ISP63**LasToscas

Y de tecnicatura:.

https://www.facebook.com/IspN4AnexoLasToscas/

Otros datos para agendar

Horarios de atención: de 18hs a 23hs

Dirección: Sede Central:

Calle 19 Nº 411.Las Toscas - (Santa Fe.)

Teléfono: 03482-492041

E-mail de consultas:

Del instituto: instituto63ltoscas@gmail.com

De Tecnicatura: tecnicatura.anexolt@gmail.com

CONOCIENDO LA CARRERA

- Denominación de la Carrera:

Técnicatura Superior en Desarrollo de Software.

- Familia profesional: *Informática.*
- Título a otorgar: *Técnico Superior en Desarrollo de Software.*
- Duración de la carrera en años académicos: *3 (tres) años.*
- Condiciones de ingreso*: Estudios Secundarios Completos.*

El Técnico Superior en Desarrollo de Software, (también denominado analista programador o programador) será capaz de utilizar las herramientas informáticas existentes y hacerlas funcionar de manera eficaz y eficiente, contemplando en su formación distintas áreas de conocimiento, entre las cuales se encuentran:

Programación, metodologías de desarrollo, arquitectura y redes, diseño de sistemas, bases de datos, dirección de proyectos informáticos, siendo imprescindible que en su formación profesional adquiera las capacidades para adaptarse a los cambios constantes en la materia, con un perfil creativo e innovador y con mentalidad de trabajo en equipo. Con esta formación se encontrará capacitado para producir artefactos de software, lo que comprende su diseño detallado, construcción y verificación unitaria, así como su depuración, optimización y mantenimiento; desarrollando las actividades descriptas en el perfil profesional y cumpliendo con los criterios de realización establecidos para las mismas en el marco de un equipo de trabajo organizado por proyecto. Entendiendo como artefacto de software cualquier parte del software (es decir modelos/descripciones) desarrollado y utilizado durante el desarrollo y mantenimiento de software, tales como: modelos de arquitectura y de diseño, código de fuente y ejecutable (programas), instrucciones de configuración, datos de prueba, scrips de prueba, modelos de proceso, planes de proyecto, u otra documentación pertinente.

Finalidades Formativas:

Desde esta perspectiva se definen las siguientes finalidades formativas que tienden a formar técnicos superiores con capacidad para:

- Desarrollar una formación técnica y profesional específica para producir artefactos de software con todo lo que ello comprende.
- Reconocer las prácticas y los procedimientos comunes en los entornos organizacionales para favorecer la administración de la información y el desarrollo de software.
- Desarrollar habilidades que integren conocimientos teórico-prácticos, capacidad de análisis crítico, resolución de problemas y toma de decisiones en contextos complejos de incertidumbre; que comprende a la sociedad como una construcción humana dotada con el tiempo, el espacio y la historia.
- Operar de modo amplio y autónomo en el ámbito local y regional a través de la formación en el área de desarrollo de software.
- Gestionar saberes que permitan adaptarse a los rápidos adelantos de las tecnologías de la información y la comunicación y actuar con flexibilidad y disposición para aprender a aprender durante toda la vida.
- Lograr una actitud ética y preparación para ser ciudadano activo, responsable y comprometido con la realidad, entendiendo y atendiendo a las demandas y necesidades del contexto socio productivo en el cual se desarrolla, aplicando las normas de sostenibilidad ambiental, con una mirada integral y cuidadosa del medio ambiente.
- Desarrollar habilidades comunicativas, sociales y laborales que favorezcan el trabajo en equipo, promuevan la motivación y liderazgo a nivel personal y también se orienten al logro de los objetivos de la organización.
- Desarrollar capacidades emprendedoras requeridas para ser protagonistas de procesos de cambio, dirigidos a mejorar la empleabilidad, la productividad y la construcción de sus propios proyectos ocupacionales.

PERFIL PROFESIONAL DEL TÉCNICO SUPERIOR EN DESARROLLO DE SOFTWARE

El Técnico Superior en Desarrollo de Software está capacitado para

- Manifestar conocimientos, habilidades, destrezas, valores y actitudes en situaciones reales de trabajo, conforme a criterios de profesionalidad propios de su área y responsabilidad social.
- Producir artefactos de Software (programas, módulos, objetos), lo que comprende su diseño detallado, construcción - reutilizando elementos existentes o programándolos enteramente- y verificación unitaria, así como su depuración, optimización y mantenimiento; desarrollando las actividades descriptas en el perfil profesional y cumpliendo con los criterios de realización establecidos para las mismas en el marco de un equipo de trabajo organizado por proyecto.
- Conformar equipos de Proyecto. El proceso de desarrollo de software es una tarea grupal, o individual y multidisciplinaria que se organiza por proyectos. Cada proyecto es negociado y acordado con el cliente o usuario y llevado a cabo por un equipo de trabajo constituido "ad-hoc", conducido y administrado por un líder que mantiene la relación diaria con el cliente o usuario y asume la responsabilidad operativa del proyecto.
- Diseñar en forma detallada la parte del software que les correspondiere (A partir de especificaciones de diseño y del conocimiento de la arquitectura del sistema), la construyen, preferiblemente en base a artefactos de software ya existentes y adaptando o escribiendo lo que sea necesario, así como documentándola para facilitar su posterior mantenimiento por otros.
- Verifican unitariamente lo producido y lo entregan para ser probado e integrado al resto.

El Técnico Superior en Desarrollo de Software participa en proyectos de desarrollo de software desempeñando roles que tienen por objeto producir artefactos de software. Estos artefactos suelen integrarse en aplicaciones o subsistemas que interactúan entre sí, con otras aplicaciones ya existentes desarrolladas con la misma o distinta tecnología, con el sistema operativo del computador u otro software de base (motor de base de datos, navegador, monitor de comunicaciones) configurando distintas capas de software que

pueden estar distribuidas en diversas máquinas situadas en la misma o distintas ubicaciones.

La actividad del desarrollador de software, a pesar de que muchas veces se reutilicen partes ya existentes, no es rutinaria. Cada asignación representa la necesidad de dar satisfacción a determinados requisitos. Ello requiere comprender el problema y la arquitectura en la que estará inserta la solución, idear estrategias de resolución y dominar el lenguaje y ambiente de programación a emplear, así como aplicar buenas prácticas de programación, lo que incluye documentar decisiones significativas de diseño y las limitaciones que tendrá el artefacto construido.

Adicionalmente el técnico superior tiene además que lograr ciertas capacidades que resultan transversales , Entre ellas, capacidad de:

- Abstracción: implica descartar o reducir detalles poco significativos de la información sobre un problema para concentrarse en pocos elementos por vez, lo que resulta en una reducción de la complejidad, que permita conceptualizar de modo más simple un dominio de problemas para facilitar su comprensión y manejo en forma genérica de sus posibles soluciones.
- Pensamiento combinatorio: conduce a la consideración sistemática de un conjunto de alternativas, lo que incluye el manejo mental de muchas variables o detalles del problema sin perder nunca de vista el concepto o la estrategia general de resolución.
- Autorregulación: implica manejarse respetando reglas y limitaciones, tanto explícitas como implícitas, sean éstas propias o del equipo de trabajo; actuar ateniéndose a un orden propio que le facilite el acceso a lo que puede necesitar, reconocer y guardar; referenciar la información y registrarla, de tal manera que le facilite acceder posteriormente en forma rápida para evaluarla y recuperarla.
- Comunicación apropiada: implica una disposición a reconocer que existen otros que pueden aportar información útil o a quienes pueda interesar lo que hace.
- Supone reconocer su rol y el de cada integrante del proyecto, transmitir la información necesaria en forma precisa y en un lenguaje apropiado para el entendimiento mutuo en interacciones individuales o grupales, o en forma escrita, utilizando, si es necesario para ello, el idioma inglés, que debe interpretar con propiedad a nivel técnico.
- Trabajo en equipo: implica adoptar una actitud abierta, estar dispuesto a compartir información y conocimientos, a tomar en cuenta a los destinatarios del producto que está construyendo, a brindar, pedir y aceptar ayuda cuando ésta resulte necesaria para facilitar su propia labor o la de otro integrante del equipo.

Además, se requiere:

- Actitud de aprendizaje permanente: implica aprender a capitalizar experiencias a partir de su propio trabajo, a tomar iniciativas para actualizar o profundizar sus conocimientos y habilidades, investigar fuentes de información o herramientas que le puedan ser útiles.
- Actitud ética: implica el ejercicio profesional respetando principios éticos y adecuación al marco legal, como así también conocer y aplicar la normativa legal vigente.

FUNCIONES QUE EJERCE EL PROFESIONAL

A continuación se presentan funciones del perfil profesional del Técnico Superior en Desarrollo de Software, de las cuales se pueden identificar las actividades profesionales:

"Modelizar artefactos de software a partir de especificaciones, refinándolas en caso necesario, para determinar el diseño detallado y las características de una solución que las satisfaga en el contexto de la arquitectura del sistema de software del cual van a formar parte"

"Construir los artefactos de software que implementen el diseño realizado, aplicando patrones o reutilizando código en la medida en que resulte posible. Al hacer esto, y en función de lo acordado para el proyecto, optimizará el desempeño de lo que construya aplicando buenas prácticas de programación y documentación".

"Verificar los artefactos de software construidos considerando las necesidades de cobertura de la prueba. Para ello diseña los casos considerando el entorno de pruebas y ejecuta pruebas unitarias, así como registra los datos y resultados. De ser necesario, realiza acciones correctivas sobre el código hasta asegurarse de que cumpla con las especificaciones recibidas".

"Revisar el código de artefactos de software para resolver defectos o mejorarlo. Este código puede ser propio o ajeno. Esta actividad comprende revisiones cruzadas con otros integrantes del proyecto para asegurar la calidad del producto. Algunas asignaciones requieren una revisión de código ya existente para poder ampliar funcionalidades o refactorizarlo".

"Documentar sus actividades y los resultados obtenidos aportando elementos para asegurar la calidad de los proyectos de acuerdo con normas y estándares establecidos".

"Gestionar sus propias actividades dentro del equipo de trabajo del proyecto. Ello

comprende la planificación (organización y control) de las tareas a realizar, el oportuno reporte de avances y dificultades y el registro y reflexión sobre lo realizado para capitalizar experiencias y estimar métricas aplicables a su actividad".

"Interactuar con los diferentes roles ocupacionales y áreas organizacionales, mediante un trabajo en equipo de carácter cooperativo, con capacidad para negociar, argumentar y articular propuestas, necesidades y expectativas".

"Generar propuestas innovadoras y/o emprendimientos productivos propios del ámbito del desarrollo de software"

Área Ocupacional

Este técnico se ocupa en organizaciones de diversos tipos. Empresas que realizan desarrollo de software por encargo de organizaciones locales o extranjeras, que proveen software junto con otros servicios de asesoramiento y consultoría, y, en menor número, que desarrollan sus propios productos de software para vender en el país o en el exterior.
También en organizaciones dedicadas a otras actividades, pero que producen el software que necesitan para desarrollar sus propias actividades o que integran en productos que venden.
Su posición ocupacional suele denominarse analista programador o programador, aunque últimamente se está generalizando una denominación más abarcativa y menos categorizante, desarrollador de software.
Integra equipos de proyecto dedicados al desarrollo o mantenimiento de software y recibe asignaciones específicas que tiene que resolver en lapsos que suelen medirse en términos de días o semanas, produciendo artefactos que satisfagan especificaciones y se integren al sistema objeto del proyecto.
Resuelve estas asignaciones individualmente o trabajando en pares, recibiendo la supervisión asesoramiento de un líder de proyecto o de grupo, con quien consulta dudas y decisiones significativas o comunica inconvenientes. También recibe apoyo y brinda colaboración a otros miembros del grupo. Su trabajo es verificado por un grupo de "testing" y eventuales controles cruzados de código importante. Con una mayor experiencia o especialización en determinadas tecnologías o metodologías, posibles evoluciones ocupacionales del Técnico Superior en Desarrollo de Software son el liderar grupos de trabajo o asumir roles de analista técnico en la materia de su especialidad.
Asimismo, puede desempeñarse en forma autónoma, asumiendo la mayor parte de las tareas propias del proceso, sobre todo trabajando en forma independiente resolviendo problemas de pequeñas organizaciones que requieren sistemas de baja complejidad y reducida dimensión.
Por otra parte, Técnicos Superiores en Desarrollo de Software o profesionales equivalentes con capacidad emprendedora pueden y suelen asociarse entre ellos para generar sus propias empresas para brindar servicios de desarrollo y proveer software a terceros.

PLAN DE ESTUDIOS DE LA CARRERA

PRIMER AÑO

Campos	Unidades Curriculares	Año	Régimen	HCS
FG	Comunicación	1	Cuatr. 1	3
	Unidad de Definición Institucional I	1	Cuatr. 2	3
FF	Matemática	1	Anual	4
	Inglés Técnico I	1	Anual	3
	Administración	1	Anual	3
FE	Tecnología de la Información	1	Anual	3
	Lógica y Estructura de Datos	1	Anual	4
	Ingeniería de Software I	1	Anual	4
	Sistemas Operativos	1	Anual	4
			Total Horas Cátedra	28

SEGUNDO AÑO

Campos	Unidades Curriculares	Año	Régimen	HCS
FG	Problemáticas Socio Contemporáneas	2	Cuatr. 1	3
	Unidad de Definición Institucional II	2	Cuatr. 2	3
FF	Inglés Técnico II	2	Anual	3
	Innovación y Desarrollo Emprendedor	2	Anual	3
	Estadística	2	Anual	3
FE	Programación I	2	Anual	6
	Ingeniería de Software II	2	Anual	4
	Base de Datos I	2	Anual	4
FPP	Práctica Profesionalizante I	2	Anual	4
			Total Horas Cátedra	30

TERCER AÑO

Campos	Unidades Curriculares	Año	Régimen	HCS
FF	Ética y Responsabilidad Social	3	Cuatr. 1	3
	Derecho y Legislación Laboral	3	Cuatr. 2	3
FE	Redes y Comunicación	3	Anual	4
	Programación II	3	Anual	6
	Gestión de Proyectos de Software	3	Anual	4
	Base de Datos II	3	Anual	4
FPP	Práctica Profesionalizante II	3	Anual	6
			Total Horas Cátedra	27

FF: Formación de Fundamento, *FE: Formación Especifica , *FPP: Formación de la Práctica Profesionalizante, * HCS: Horas Cátedra Semanales

Régimen de Correlatividades

La trayectoria que realice cada estudiante en la carrera, deberá respetar las siguientes pautas del régimen de cursado y correlatividades. Las correlatividades se establecen en función de los procesos que se pretenden desarrollar en el transcurso de la formación y de los alcances de contenidos correspondientes a cada unidad curricular

Para Rendir	Debe tener aprobada
Inglés Técnico II	Inglés Técnico I
Programación I	Lógica y Estructura de Datos
Ingeniería de Software II	Ingeniería de Software I
Redes y Comunicación	Tecnología de la Información Sistemas Operativos
Programación II	Programación I
Bases de Datos II	Bases de Datos I Sistemas Operativos
Gestión de Proyectos de Software	Ingeniería de Software II
Práctica Profesionalizante II	Práctica Profesionalizante I Administración Innovación y Desarrollo Emprendedor

REGLAMENTO ACADÉMICO MARCO

El Instituto se rige por el Reglamento Académico Marco (RAM), te invitamos a leerlo siguiendo este link:

http://ispn4-santafe.edu.ar/Informacion/Dec4199-15RAM.pdf

El Reglamento Académico Marco (RAM) Decreto n° 4199/15, NORMA todos los Institutos de Educación Superior de la Provincia de Santa Fe, de **Gestión Estatal y Privada, y Regula: Ingreso, Trayectoria Formativa, Permanencia y Promoción** de los estudiantes; y la **Formación Continua de los Egresados**

Defines algunas **cuestiones importantes** como las siguientes:

- Las instituciones formadoras promueven la flexibilidad en los trayectos académicos brindando al estudiante posibilidades de selección personal de recorridos formativos.
- A partir del año posterior al Ingreso los estudiantes deberán inscribirse en las Unidades Curriculares que deseen, eligiendo condición y modalidad. La inscripción se llevará a cabo en dos fechas por año académico.
- Estudiante regular de la carrera debe regularizar o aprobar una Unidad Curricular por año calendario.
- Sistema de calificación decimal de 1 (uno) a 10 (diez) puntos. La nota mínima de aprobación: 6.
- Los estudiantes podrán elegir condición y modalidad para cursar las Unidades Curriculares.
- Condición de regular: modalidad de cursado *presencia*l o cursado *semi-presencia*l, o *libre*.
- Los estudiantes deberán inscribirse a cada Unidad Curricular optando por la condición y modalidad.
- Los estudiantes inscriptos como regulares con cursado presencial o semi-presencial, que una vez comenzado el período de clases, no pudieren reunir las condiciones exigidas por la modalidad de su elección por razones personales y/o laborales u otras debidamente fundamentadas, podrán cambiarse a las de regular con cursado semi-presencial o libre según sea el caso.
- El estudiante tendrá derecho a recuperatorios en todas las instancias acreditables (parciales, trabajos prácticos, coloquios, trabajos de campo, otros que determinen los docentes en sus planificaciones).
- La asistencia se computará por cada Unidad Curricular y hora de clase dictada.
- Solo podrán cursar en condición de libre las Unidades Curriculares con formato materia.

Tipos de Cursados y Promociones:

	ASISTENCIA	TRABAJOS PRÁCTICOS	PARCIALES	ACREDITACIÓN
REGULAR CON PROMOCIÓN DIRECTA	**75%**	**100%**	**APROBADOS CON NOTA 8.**	**COLOQUIO INTEGRADOR CON NOTA 8 MÍNIMO.**
REGULAR PRESENCIAL	**75% - 50% razones laborales, salud u otros.**	**75%**	**APROBADOS CON NOTA 6. RECUPERATORIOS.**	**EXAMEN FINAL CON TRIBUNAL EXAMINADOR**
REGULAR SEMI PRESENCIAL	**40%**	**100%**	**APROBADOS CON NOTA 6. RECUPERATORIOS**	**EXAMEN FINAL CON TRIBUNAL EXAMINADOR**
LIBRE	**……**	**……**	**……**	**EXAMEN FINAL CON TRIBUNAL EXAMINADOR**

- Estudiante libre deberá aprobar un examen final ante un Tribunal con una nota mínima de 6 (seis) puntos.
- Cada profesor adjuntará a su planificación el programa de examen correspondiente, indicando la modalidad y bibliografía.
- La regularidad tendrá validez durante 3 (tres) años consecutivos a partir del primer turno correspondiente al año lectivo siguiente al de la cursada.
- Cuando haya más de un llamado por turno el estudiante podrá presentarse en todos ellos.

De los Exámenes Finales

- **Examen final**: regulares o libres, *deberán inscribirse* para acceder al mismo. La modalidad de los exámenes finales podrá ser oral, escrito, de desempeño o mixta
- La **nota de aprobación**: del examen final, o la del promedio de los exámenes finales cuando se hayan combinado las modalidades: 6 (seis) o más sin centésimos.
- El examen final se realiza ante un **Tribunal o Comisión evaluadora** formada por 3 (tres) miembros, el profesor de la Unidad, quién oficiará de Presidente de mesa y 2 (dos) profesores de Unidades Curriculares afines.

Reglamento Académico Marco

Promoción Directa: no rendir un examen final.

	ASISTENCIA	TRABAJOS PRÁCTICOS	PARCIALES	ACREDITACIÓN
REGULAR CON PROMOCIÓN DIRECTA	75%	100%	APROBADOS CON NOTA 8.	COLOQUIO INTEGRADOR CON NOTA 8 MÍNIMO.

Si el alumno regulariza pero no la promociona, tiene derecho a un examen final escrito, en cualquiera de los turnos correspondientes, debiendo obtener una calificación mínima de 6 (seis). Mantiene la regularidad durante 3 (tres) años consecutivos a partir del primer turno correspondiente al año lectivo siguiente al de la cursada

Mesas de Exámenes Especiales

Podrán ser solicitadas:

a) Cuando adeudare hasta dos materias del último año de la carrera para finalizar sus estudios.

b) Cuando, por razones de enfermedad, causa grave de índole familiar, laboral u otros motivos debidamente justificados, fehacientemente comprobados en tiempo y en forma, no pudiera presentarse a rendir examen en la fecha y horario estipulado, hasta tres veces a lo largo de su carrera.

c) Cuando caducaren los Planes de Estudio con los cuales cursó.

Seminarios

- Podrán ser **cursados** solamente con categoría de estudiantes regulares, ya sea con cursado *presencial* o *semi-presencial*.
- **Aprobación**: se debe exigir la presentación de un trabajo final de escritura académica(monografías, ensayos, proyectos, entre otros) con su correspondiente defensa oral. La nota de aprobación: 6 (seis).
- La regularidad tendrá **validez** durante un año a partir del primer turno de examen del siguiente al de la cursada.

Talleres

- Sólo admitirán el cursado regular **presencial**.
- **75% de asistencia** a las clases áulicas.
- **Aprobar el 100% de las instancias de evaluación** previstas en la planificación anual, contemplando una instancia final de integración. La nota será de 6 (seis) o más sin centésimos.
- El estudiante que no haya aprobado podrá presentarse hasta **dos turnos consecutivos inmediatos posteriores** a la finalización de la cursada.

Homologaciones y Pases

- **HOMOLOGACIONES**: El criterio a aplicarse será el de la vigencia de los contenidos y la bibliografía, y no el de su antigüedad. Podrán concederse homologaciones parciales: cada institución indicará los pasos a seguir y requisitos a cumplimentar por parte del solicitante.

- **PASES:** Estudiantes de Nivel Superior que cambien de Instituto pero continúen con la misma carrera y mismo Plan. En caso de provenir de la misma carrera pero con distinto plan de estudios: se tramitan homologaciones.

SÍNTESIS EXPLICATIVA DE LOS ESPACIOS CURRICULARES DE 1er AÑO

Comunicación

Finalidad Formativa

Esta unidad curricular propone trabajar las experiencias del habla, la escucha, la lectura y la escritura para que el estudiante pueda plasmar sus ideas y proyectos en los ámbitos tanto personal como laboral, en un proceso de constitución subjetiva, para lograr soltura y solvencia tanto en sus discursos y planteos teóricos como en la elaboración de informes.

Ejes de Contenido

El habla, la escucha, la lectura y la escritura como experiencias en la comunicación. Aportes teórico- metodológicos. Diferencias entre oralidad y escritura. Los conceptos de comunicación verbal y no verbal. Los diferentes tipos y elementos de comunicación. Los conceptos de información, expresión y comunicación. Las variables lingüísticas. Metalenguaje. El proceso de expresión y comunicación oral. Expresión y comunicación. El circuito del habla. Lenguaje corporal. La comunicación eficaz y las técnicas de oratoria. El dialogo, el debate, la exposición, la recepción. La gestualidad y la puesta en escena. El discurso persuasivo. Tipos de audiencia. La comunicación verbal y el registro escrito. El proceso de escritura y la textualidad. El proceso de escritura y las formas discursivas. La redacción. La narración. La argumentación. La comunicación profesional y sus instrumentos. Narrativas transmedia. Contexto, situación comunicativa e intencionalidad. Elaboración de informes. Presentaciones laborales.

Matemática

Finalidad Formativa

Esta unidad curricular permite introducir a los estudiantes en los conceptos básicos, con el propósito de desarrollar la capacidad de razonamiento y de resolución de problemas para fortalecer las bases necesarias para el pensamiento computacional.

Está destinado a abordar saberes científico-tecnológicos que otorgan sostén a los conocimientos, habilidades, destrezas propios del campo profesional.

SÍNTESIS EXPLICATIVA DE LOS ESPACIOS CURRICULARES DE 1er AÑO

Ejes de Contenido

Funciones, tipos: inyectivas, sobreyectivas, inversas, composición. Relaciones, tipos: reflexividad, simetría, transitividad, equivalencia. Conjuntos; diagramas de Venn, operaciones, complementos, producto cartesiano, conjunto potencia. Numerabilidad y cardinalidad. Aritmética modular. Relaciones de confruencia. Sistemas de numeración. Elementos de lógica. Lógica proposicional, conectivos lógicos. Tablas de verdad. Formas normales; conjuntiva y disyuntiva. Validez. Lógica de predicados; cuantificadores universal y existencial. Modus ponens y modus tollens. Limitaciones de la lógica de predicados.

Técnicas de demostración. Nociones de implicación, conversa, inversa, contrapositivo, negación y contradicción. La estructura de las demostraciones matemáticas. Demostración directa, por contraejemplo, por contradicción. Inducción matemática. Inducción fuerte. Definiciones matemáticas recursivas. Buen ordenamiento.

Inglés Técnico I

Finalidad Formativa

Esta unidad curricular permite al estudiante desarrollar la competencia lectora posibilitando la autonomía en la lectura e interpretación de textos técnicos y reconocer las formas lingüísticas del discurso escrito en su función comunicativa, a través del acceso a bibliografía en inglés en el área del desarrollo de software.

Ejes de Contenido

Lectura e interpretación de textos e información técnica en inglés. Comprender textos de complejidad creciente en inglés, para comunicarse solicitando o aportando información técnica por email o en foros y listas de discusión.

Administración

Finalidad Formativa

Esta unidad curricular permite al estudiante reconocer a la Administración como una disciplina social, adquirir el manejo preciso de los conceptos y técnicas que le permitan obtener la competencia necesaria para poder desempeñarse en las diferentes

SÍNTESIS EXPLICATIVA DE LOS ESPACIOS CURRICULARES DE 1er AÑO

organizaciones, adaptando su trabajo a los cambios que puedan originarse en ella y/o en el entorno.

Ejes de Contenido

Elementos de teoría general de los sistemas, enfoque sistémico de la organización. Elementos de estructura y comportamiento de las organizaciones, organización estructurada por funciones o líneas de productos, el manejo de sedes.

Concepto de proceso. Procesos del ciclo de ventas y cobranzas; del ciclo de compras y pagos. Nociones de proceso de gestión y transformación de materiales y su organización. Comprobantes usuales, requerimientos legales y fiscales. Concepto de recurso y su gestión. El papel de los sistemas de información en la organización. Nociones de control interno. La contabilidad como sistema de información. Algunas características de organizaciones y procesos de servicio.

Los niveles de la administración: la planificación estratégica, el control de gestión, el control operativo y el tipo de sistemas de información asociados a los mismos.

Conceptos de planificación. Descomposición de pequeños proyectos en planes de trabajo con actividades específicas. Herramientas de modelado de procesos administrativos (organigrama, fluxogramas, otros.) Secuenciación de actividades y estimación de tiempos, métodos de planificación: Gantt, camino crítico. Coordinación de actividades a realizar por otros. Resolución de conflictos surgidos por la necesidad de compartir recursos. Necesidad de registrar y documentar.

El rol de la información en las organizaciones. La importancia de la tecnología de información en el mundo globalizado. Los factores organizacionales y gerenciales en los sistemas de información. El impacto de los SI en las organizaciones. Tipos de sistemas de información.

Tecnología de la Información

Finalidad Formativa

Esta unidad curricular permite al estudiante reconocer el rol de la información en las organizaciones y la importancia que la tecnología de información tiene en el mundo

SÍNTESIS EXPLICATIVA DE LOS ESPACIOS CURRICULARES DE 1er AÑO

globalizado, donde a partir de un desempeño como profesional calificado pueden detectarse y analizarse problemas, proponiendo la mejor solución sistémica.

Ejes de Contenido

Conceptos de tecnología de la información, evolución histórica, tecnologías que la integran, disciplinas que forman parte (ciencia de la computación, ingeniería de software, sistemas de información, ingeniería en computación) o aportan a la misma. El problema de la complejidad. Concepto de computación paralela, concurrente, multinúcleos. Evolución del computador, su organización y unidades funcionales que lo componen. Arquitectura interna de computadores, unidad central de procesamiento, instrucciones y flujo de la información. Tipos y niveles de organización de la memoria interna y externa (sistemas de memoria, tecnologías y jerarquías, memoria caché, memoria virtual, dispositivos de almacenamiento secundario). Periféricos, clasificación y utilización. Funcionamiento del programa a nivel de la máquina (principalmente como medio de comprender características de su funcionamiento).

Introducción a la lógica digital, compuertas lógicas, flip-flops, circuitos. Expresiones lógicas y funciones booleanas. Representación de datos numéricos, aritmética con y sin signo, concepto de overflow. Rango, precisión y errores en aritmética de punto flotante. Representación de caracteres, audio e imágenes. Compresión de datos.

Orígenes y evolución de Internet y las comunicaciones digitales. Arquitecturas de red. Especializaciones en la computación y la administración de información, centrada en redes. Redes y protocolos. Computación distribuida. Paradigmas clientes/servidor y peer to peer. Computación sin cables y móvil.

Aportes de las tecnologías a la gestión de la información en las organizaciones (Data Warehousing y Data Mining, los sistemas ERP).

Aspectos técnicos de la aplicación de documento electrónico, firma digital, comercio electrónico y gobierno electrónico, en las organizaciones.

SÍNTESIS EXPLICATIVA DE LOS ESPACIOS CURRICULARES DE 1er AÑO

Lógica y Estructura de Datos

Finalidad Formativa

Esta unidad curricular permite, en primer lugar introducir a los estudiantes en los conceptos básicos, para luego abordar con solvencia los saberes científicos y tecnológicos que otorgan sostén a los conocimientos, habilidades, y destrezas como instrumentos para el desarrollo de la capacidad de razonamiento y de resolución de problemas para fortalecer las bases necesarias para el pensamiento computacional.

Ejes de Contenido

Concepto de algoritmo, resolución algorítmica de problemas, estrategias de diseño, de implementación, de depuración. Algoritmos fundamentales, algoritmos numéricos simples.

Estructuras fundamentales, variables, tipos, expresiones y asignaciones, entrada/salida, estructuras de control condicionales e iterativas, funciones y pasaje de parámetros, descomposición estructurada.

Concepto de lenguaje de alto nivel y la necesidad de traducción, comparación entre compiladores e intérpretes, aspectos de la traducción dependientes y no dependientes de la máquina. Programas generadores de código.

Verificación unitaria de unidades de código, concepto de cubrimiento, organización, ejecución y documentación de la prueba.

Representación de datos numéricos, rango, precisión y errores de redondeo. Arreglos. Representación de datos de caracteres, listas y su procesamiento. Manejo de memoria en tiempo de ejecución, punteros y referencias, estructuras encadenadas, pilas, colas y tablas de hashing. Recolección de espacios no utilizados. La elección de una estructura de datos adecuada.

Paradigma funcional, lógico, imperativo, características fundamentales.

Archivos secuenciales y de acceso directo. Organización y acceso. Registros. Administración. Operaciones básicas. Procesamiento en memoria secundaria.

Recursividad. Manejo de excepciones.

SÍNTESIS EXPLICATIVA DE LOS ESPACIOS CURRICULARES DE 1er AÑO

Ingeniería de Software I

Finalidad Formativa

Esta unidad curricular permite introducir al estudiante en el trabajo de Ingeniería, llevando a cabo proyectos con la utilización de métodos y la ayuda de herramientas propias de su ámbito de acción. Reconocer, diferenciar, documentar y validar las necesidades que justifican sus proyectos. Documentar escenarios y propuestas que respondan a dichas necesidades.

Ejes de Contenido

Ingeniería del Software: conceptos generales. Paradigmas, métodos y herramientas: una visión global. Modelos de desarrollo de Software.

Análisis de requerimientos de software, el proceso, partes interesadas. Requerimientos funcionales, prioridades y criterios de realización. Análisis orientado a objetos y UML. Diagramas de clase. Escenarios, historias y casos de uso; diseño centrado en el usuario. Representación del comportamiento: diagramas de secuencia, máquinas de estado, diagramas de actividad. Redes de Petri. Pre y post condiciones.

Análisis de datos: datos de referencia y de operaciones; datos de nivel de recursos y de volumen de actividad. Modelo Entidad/Relación. Principales formas normales. Diccionario de datos.

Requerimientos no funcionales, ejemplos y su influencia en el diseño del software. Validación de requerimientos. Estándares de documentos de requerimientos. Herramientas de modelización. UML.

Ambientes gráficos para edición, editores inteligentes. Herramientas integradas disponibles para la edición en distintos lenguajes y ambientes. Bibliotecas de clases, programas y rutinas.

Sistemas Operativos

Finalidad Formativa

Esta unidad curricular permite conocer sobre sistemas operativos desarrollando la capacidad de manejarse hábilmente con diversos editores, configurar aspectos de software y hardware y explotar con habilidad recursos de programación y servicios que

SÍNTESIS EXPLICATIVA DE LOS ESPACIOS CURRICULARES DE 1er AÑO

los Sistemas operativos ofrecen, incluyendo entre los mismos bibliotecas de objetos y programas propios, de su organización o disponibles a través de Internet, así como plantear y resolver consultas de problemas de programación a través de foros y listas públicas o privadas.

Ejes de Contenido

Los sistemas operativos, su papel y propósito, la historia de su desarrollo, funcionalidades típicas. Mecanismos que soportan los modelos cliente-servidor y otros dispositivos. Características y objetivos de su diseño y su influencia en la seguridad, interoperabilidad, capacidad multimedial.

Estructuras de sistemas operativos (monolíticos, modulares y de "micro kernel"). Abstracciones, procesos y recursos. Organización de los dispositivos, interrupciones: métodos e implementación. Concepto de estados usuario/supervisor y protección, transición al modo supervisor.

Estados y transiciones; cola de procesos, bloque de control de procesos. Despacho, "switching" de contexto, "switching" cooperativo y "preempted". Ejecución concurrente: ventajas y desventajas. El problema de la exclusión mutua y algunas soluciones. Bloqueos: causas, condiciones, prevención. Paso de mensajes sincrónico y asincrónico. Problema de consumidor-productor y sincronización (mutex, semáforos). Multiprocesamiento (interrupción de ciclos, reentrada).

Políticas de despacho de procesos; programación con y sin prioridades de interrupción. Procesos y "threads". Elementos de tiempo real y tiempos límite.

Administración de memoria. Revisión de memoria física y hardware de administración de memoria. Paginamiento y memoria virtual. "Working sets" y "trashing". "Cacheo".

Administración de dispositivos, características de dispositivos seriales y paralelos. Abstracción de diferencias entre dispositivos. Estrategias de "buffering". Acceso directo a memoria. Recuperación de fallas.

"Scripting". Comandos básicos del sistema, creación de "scripts", pasaje de parámetros. Ejecución de un "script".

Seguridad y protección. Políticas y mecanismos de separación. Métodos y dispositivos

SÍNTESIS EXPLICATIVA DE LOS ESPACIOS CURRICULARES DE 1er AÑO

de seguridad. Protección, control de acceso y autenticación. Backups.

Sistemas de archivo (datos, metadatos, operaciones, organización, “buffering”, secuenciales y no secuenciales). Índices: contenido y estructura. Técnicas estándares de implementación. Archivos de mapeo de memoria. Sistemas de archivo para propósitos especiales. Denominación, búsqueda, acceso, backups.

Máquinas virtuales, concepto, jerarquía de máquinas virtuales, lenguajes intermedios, asuntos de seguridad que surgen al ejecutar código en una máquina diferente.

Unidad de Definición Institucional:

Mundo del Trabajo: Subjetividad y Organización.

Las unidades de definición institucional se seleccionan por institución y por carrera de acuerdo a las prioridades de los contextos sociales y culturales. Derivan de un listado de problemáticas ofrecidas por la jurisdicción , de

dictado cuatrimestral y se incluyen en el campo de la Formación General.

CONTENIDO DISCIPLINAR: Comunicación

LAS CUATRO DESTREZAS: EXPRESIÓN ORAL

Este texto es fundamentalmente un resumen de:
Cassany, Daniel; Marta Luna; Glòria Sanz (1994) [reimpresión 2008]: Enseñar lengua. Barcelona: Graó, § 6.3. Expresión oral, págs. 134-150.

1. La necesidad de "enseñar a hablar"

En la sociedad actual, "saber hablar" es una necesidad. Constantemente necesitamos un nivel de expresión oral igual de alto que el de expresión escrita: es preciso realizar exposiciones orales en clase, realizar entrevistas de trabajo, dialogar por teléfono con desconocidos, dejar mensajes en un contestador automático, realizar reuniones por Skype, etc. Una persona que no puede expresarse oralmente de manera coherente y clara limita gravemente sus posibilidades personales y profesionales. Una de las tareas de todo docente es hacer ver a los alumnos la relevancia de lo oral en la vida cotidiana y la importancia de tratar esta destreza explícitamente en el aula.

Evidentemente, no se enseña a hablar desde cero. Los alumnos ya son capaces de participar en las interacciones orales que se producen en sus situaciones cotidianas: conversaciones familiares y coloquiales, explicaciones breves, etc. Lo que es necesario es ampliar el abanico expresivo del alumno, del mismo modo que se amplía su conocimiento del medio o su preparación física o plástica (que ya se ha iniciado antes de llegar a la escuela). Así, habrá que trabajar explícitamente comunicaciones de ámbito social: exposiciones, debates públicos, conversaciones telefónicas con desconocidos, intervenciones en reuniones, etc.; y también comunicaciones académicas: entrevistas, exámenes orales, exposiciones, etc. Algunos libros donde se hacen propuestas al respecto son Badia y Vila (1993), Coromina (1984), García et al. (1986), Saló (1990), Solé (1988), entre otros muchos.

Mención especial merece la enseñanza de la expresión oral en las lenguas propias (catalán, gallego, euskera) de las comunidades bilingües. Las leyes y los currículos escolares proponen que, al acabar su escolarización, el alumnado domine con una misma capacidad suficiente las dos lenguas cooficiales de la comunidad (las arriba mencionadas y el castellano). Para alcanzar este objetivo, es necesario el trabajo explícito de la comunicación oral en catalán, gallego y vasco (y no solo de la expresión escrita), de forma pareja a la instrucción formal de la expresión oral en castellano (muy especialmente en el caso de los alumnos cuya lengua materna es el castellano y han de aprender las lenguas propias de cada zona).

2. Tipos de texto y necesidades orales

Para determinar cuáles son las necesidades de expresión oral de los alumnos, comencemos analizando los distintos tipos de situaciones comunicativas orales que pueden darse. Una de

CONTENIDO DISCIPLINAR: Comunicación

las tipologías existentes, distingue entre situaciones comunicativas orales autogestionadas y plurigestionadas. En las situaciones autogestionadas, los receptores no tienen la posibilidad inmediata de responder y, por tanto, de participar como emisores. En las situaciones plurigestionadas, los interlocutores adoptan alternativamente los roles de emisor y receptor. Las primeras, por tanto, requieren la capacidad de preparación y autoregulación del discurso, mientras que las segundas ponen énfasis en la interacción y colaboración comunicativa. El siguiente cuadro recoge las principales diferencias. Dependiendo del tema, los interlocutores, el medio, etc. los textos orales de una u otra modalidad reflejaran un estilo de lengua formal o coloquial, en distintos grados.

COMUNICACIÓN ORAL

autogestionada exposición, conferencia, charla, discurso	*plurigestionada* diálogo, tertulia, entrevista, conversación, debate
1. Una sola persona elabora el texto. Hay una sola voz.	1. Varias personas colaboran en la gestión del texto. Varias voces.
2. El emisor gestiona el texto (tema, tiempo, intervención, tono, etc.)	2. Los interlocutores negocian el texto (tema, intervenciones, tono, etc.)
	3. Se establecen turnos de palabra, hay intercambio de roles de emisor-receptor, encabalgamientos de intervenciones, etc.
4. Modalidad básicamente enunciativa: afirmaciones.	4. Cambios frecuentes de modalidad: preguntas, respuestas, negaciones, afirmaciones, etc.

Una vez presentada esta tipología, debe responderse la siguiente pregunta ¿cuáles de estas situaciones de comunicación oral deben trabajarse en clase? ¿qué necesidades de lengua tienen los alumnos? Los diversos tipos de comunicación oral deben tener su lugar en el espacio de clase dedicado a la comunicación oral. Por una parte, es evidente que las situaciones comunicativas autogestionadas (por ejemplo, exposiciones en público) son las más alejadas de la vida cotidiana de los alumnos, y deben trabajarse explícitamente. Pero, por otra parte, muchos alumnos tienen problemas en los diálogos que se establecen en clase, por ejemplo en el trabajo en grupo (hay alumnos que no escuchan a sus compañeros, que monopolizan la palabra, que tienen una pobre expresión, que se inhiben y no dicen nada, que tienen “salidas de tono”, etc.). Las situaciones orales plurigestionadas son las formas más frecuentes de comunicación humana y por ello deben también ser trabajadas explícitamente. En los niveles inicales de enseñanza habrá que poner más énfasis en los discursos plurigestionados; a medida que el alumno crece, hay que ponerle en situaciones más complejas y especiales, como las que dan lugar a situaciones de comunicación oral autogestionadas.

Por otra parte, es necesario entender que para desarrollar la expresión oral hay que trabajar tanto la corrección en la expresión como la fluidez:

Corrección	*Fluidez*
– precisión léxica	– velocidad y ritmo
– gramaticalidad	– soltura
– normativa	– seguridad
– pronunciación clara	– conexión del discurso

CONTENIDO DISCIPLINAR: Comunicación

3. Modelo teórico de expresión oral.

En este apartado se expone, desde un punto de vista teórico, en qué consiste construir un discurso oral. Bygate (1987), basándose en las situaciones de comunicación plurigestionadas, ofrece el siguiente modelo de expresión oral. Este autor distingue entre conocimientos y habilidades que se ponen en funcionamiento para construir un discurso oral. Los conocimientos son informaciones que tenemos memorizadas: incluyen el conocimiento del sistema de la lengua (gramática, léxico, etc.), pero también otros aspectos relacionados con el conocimiento del mundo (de la cultura, de las relaciones sociales, etc.).

Las segundas hacen referencia a los comportamientos que mantenemos en los actos de expresión: las habilidades de adaptarse al tema, de adecuar el lenguaje a la situación o interlocutor, etc.

MODELO DE EXPRESIÓN ORAL. Bygate 1987

A continuación se ilustrará el esquema con un ejemplo concreto. Imaginemos que tenemos que presentarlos a una entrevista de selección de personal en una organización (escuela, institución, empresa, etc.) para conseguir un trabajo. ¿Cómo funcionaría el esquema anterior?

En primer lugar, debemos decir que todos, más o menos, tenemos cierto conocimiento de este tipo de situaciones. Si hemos tenido que entrevistarnos en otras ocasiones, guardamos recuerdos y experiencias, pero también sabemos lo que socialmente se supone que ha de suceder: un diálogo entre un entrevistador y un candidato: el primero hará preguntas para explorar las capacidades para el trabajo del candidato (su experiencia laboral, formación, intereses, motivos de la solicitud, etc.) Dentro de cada cultura, todos disponemos de cierta

CONTENIDO DISCIPLINAR: Comunicación

información sobre la mayoría de las comunicaciones que se producen de forma habitual y repetida: podemos prever cómo se desarrollará una conversación en un ascensor, la conversación con un camarero, o la conversación que se producirá al comprar un coche o un vestido). Las comunicaciones humanas se estructuran y se fijan a partir de la repetición y de la experiencia que vamos adquiriendo los interlocutores. En el caso de la entrevista, todos tenemos un esquema mental de cómo sucederá: existen unos temas fijados, como acabamos de mencionar; también está establecido el modo en que se producen las intervenciones o los turnos de palabra (exposiciones breves del entrevistador, respuestas y exposiciones del candidato, etc.), y se configuran unos roles determinados (entrevistador: dirige la conversación, hace preguntas, presiona, observa, etc.; candidato: sigue el hilo propuesto por el entrevistador, responde, etc.). Rutinas es el nombre que se le da a estas estructuras comunicativas, en las que habitualmente se distingue entre la información (el contenido de

la transacción) y la interacción (estructura de las intervenciones). Las rutinas son culturales y varían de una comunidad lingüística a otra.

El conocimiento de las rutinas nos permite ejercitar la primera habilidad comunicativa, que es la planificación del discurso. A partir de la experiencia que tenemos de situaciones parecidas a la que se va a producir, podemos prever lo que pasará y decidir cómo nos comportaremos: sobre qué temas hablaremos y de qué manera. En el ejemplo de la entrevista laboral, destacaremos los puntos más relevantes de nuestro CV, y camuflaremos los posibles vacíos existentes, responderemos a las preguntas de manera completa, intentando llevar la conversación hacia los temas que dominamos, etc. Seguramente sentiremos curiosidad por el nuevo trabajo y sus características, pero no realizaremos preguntas directas al respecto.

Para ejecutar los planes desde el momento en el que empiece la entrevista, utilizaremos las microhabilidades específicas de conducción de la interacción. Por un lado hay que saber colaborar en la selección y en el desarrollo de los temas: iniciar un tema, ampliarlo, desviarlo hacia otro, acabarlo, etc. Por otro, es necesario saber dominar los turnos de palabra: saber cuándo se puede hablar, durante cuánto tiempo, y cuándo se debe ceder la palabra. Respecto a este punto, Bygate (1987) señala cinco estrategias concretas:

1. Saber indicar que se quiere hablar (gestos, sonidos, etc.)
2. Saber tomar la palabra en el momento idóneo.
3. Saber aprovechar la palabra (decir todo lo que se quiere decir, adecuarse a la estructura de las intervenciones, etc.).
4. Saber reconocer las indicaciones de los demás para tomar la palabra.
5. Saber dejar la palabra a otro.

En nuestro ejemplo, la rutina de la entrevista determina los roles del entrevistador y del entrevistado y limita, por tanto, el uso de estas estrategias. Pero basta pensar en una conversación libre entre varias personas para darse cuenta de la elegancia con la que algunas personas esquivan una pregunta, cortan una intervención o toman la palabra, mientras que otras sólo pueden dejarse llevar por los demás, o interrumpen groseramente las intervenciones de los otros.

Con el tema y los turnos de palabra acordados, los interlocutores negocian el significado. Entrevistador y candidato hablan, cada uno desde su óptica, y van adecuando lo que dicen a sus intereses y a las necesidades del otro. Es un proceso de adaptación mutua donde los dos

CONTENIDO DISCIPLINAR: Comunicación

discursos tienen que convergir. Las habilidades que se utilizan en la negociación del significado son de dos tipos: la selección del nivel de explicación y la evaluación o confirmación de la comprensión. En la selección del nivel de explicación, los interlocutores tienen que escoger el grado de detalle y de desarrollo con que se van a explicar. El defecto de información provoca incomprensión, pero el exceso resulta reiterativo y dificulta la comprensión de lo relevante. Para encontrar el grado adecuado de explicitación hay que saber qué sabe el receptor y qué le interesa. Por ejemplo, si el entrevistador pide una explicación sobre la experiencia laboral del candidato, lo más acertado es que este empiece comentando el tema de forma general y que vaya profundizando en él a medida que el otro asiente y solicita más información sobre algunos puntos concretos. Contrariamente, extenderse de entrada en una exposición detalladísima de la vida laboral desde el comienzo sería poco adecuado. En la evaluación o confirmación de la comprensión, los interlocutores confirman que el nivel de explicitación es adecuado y que se ha comprendido el mensaje. Se trata de un proceso de colaboración entre emisor y receptor, en el que ambos participan activamente. Bygate (1987) cita varias estrategias de confirmación de la comprensión:

EMISOR *yo*	RECEPTOR *tú*
– explico mi propósito por adelantado	
– te muestro cordialidad	– me muestro cordial
– evalúo la información que compartimos	
– preciso y autocorrijo mi mensaje	
– compruebo que me entiendes	– indicas que me entiendes (con gestos, asentimientos, expresión facial, etc.)
– te pido si me has entendido	– indicas lo que no entiendes, o tus dudas
– me adapto a tus indicaciones y clarifico el mensaje	– si es necesario me cortas para matizar o contrastar algún punto
– te pido tu opinión	
– resumo lo que me has dicho	

Tanto el nivel de explicitación como la negociación del significado afectan al contenido, como se ha explicado, pero también a los aspectos lingüísticos, como la selección de estructuras gramaticales y del léxico (grado de complejidad sintáctica, grado de especificidad del léxico, etc.) (selección lingüística).

El último componente de la expresión oral, que incluye habilidades específicas, es la producción real del discurso, o sea, la pronunciación de las expresiones que vehiculan los significados. En la comunicación oral, los interlocutores no siempre tienen mucho tiempo para preparar su intervención ni para procesar la intervención del otro: el emisor no puede 'preparar' detenidamente lo que dice, y el receptor no puede 'reescuchar' lo que se ha dicho. Para adaptarse a estas dificultades, los inerlocutores utilizan dos habilidades: la facilitación de la producción y la compensación de las dificultades.
En la primera, los emisores simplifican tanto como resulta posible (y adecuado a la situación e interlocutor) la estructura gramatical, hacen pausas, etc. En la segunda, los emisores ayudan al receptor a comprender lo que dicen: se autocorrigen a medida que van hablando (afinando y puliendo el significado de una expresión que pudiera no haber quedado claro), resumen lo que dicen y lo reformulan con otras palabras, indicando las ideas rpincipales, etc. Se intenta compensar las dificultades de comprensión con redundancias y repeticiones.

Finalmente, el esquema de Bygate (1987) incorpora la habilidad de la autocorrección gramatical. Es la habilidad que nos permite fijarnos en la forma del discurso y corregir posibles errores teniendo en cuenta las reglas normativas sobre gramática, léxico y pronunciación.

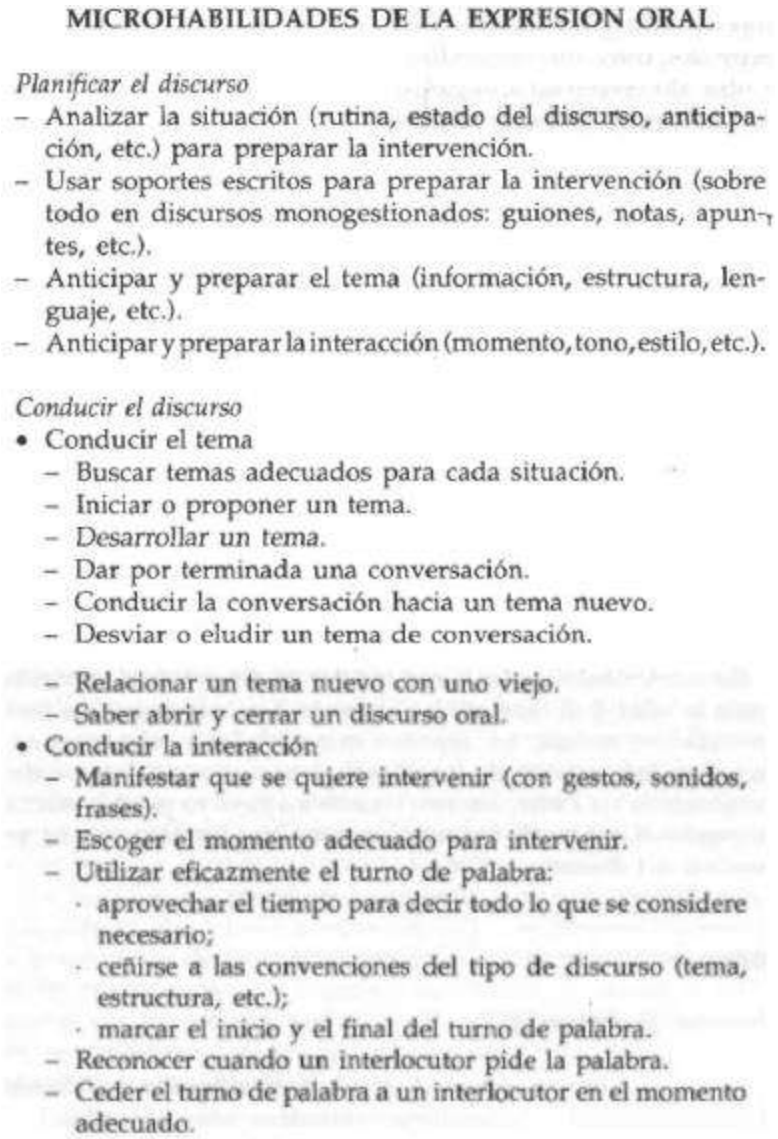
CONTENIDO DISCIPLINAR: Comunicación

Aunque se han presentado de forma lineal, este conjunto de habilidades se interrelacionan las unas con las otras durante todo el tiempo que dura la comunicación. En la entrevista, el candidato, de forma simultánea, ajustará el tema, adaptará el nivel de explicitación, controlará los turnos de habla, la corrección normativa de su intervención, etc.
Hay dos aspectos más que no se mencionan en el modelo anterior, pero que son muy importantes en la expresión oral. Se trata del control de la voz (impostación de la voz, volumen, tono, matices e inflexiones, etc.) y de la comunicación no verbal (control de la mirada, la gesticulación, el espacio entre emisor y receptor, movimiento del cuerpo, etc.).

4. Microhabilidades.

A partir del modelo teórico expuesto, se detallan en este apartado las microhabilidades involucradas en el proceso de expresión oral. La clasificación que se ofrece a continuación, establece los distintos objetivos de la expresión oral que se deben trabajar en el aula. La lista incorpora tanto las destrezas de la conversación (poligestión) como las de la exposición
oral (monogestión).

Estas microhabilidades tienen incidencia diversa en el currículum según la edad y el nivel
de los alumnos. Los más pequeños tienen necesidad de trabajar los aspectos más globales y
relevantes de la expresión (negociación del significado, interacción, evaluación de la comprensión, etc.). Los mayores pueden ya practicar aspectos como la autocorrección, producción cuidada, planificación del discurso, etc.



MICROHABILIDADES DE LA EXPRESION ORAL

Planificar el discurso
- Analizar la situación (rutina, estado del discurso, anticipación, etc.) para preparar la intervención.
- Usar soportes escritos para preparar la intervención (sobre todo en discursos monogestionados: guiones, notas, apuntes, etc.).
- Anticipar y preparar el tema (información, estructura, lenguaje, etc.).
- Anticipar y preparar la interacción (momento, tono, estilo, etc.).

Conducir el discurso
- Conducir el tema
  - Buscar temas adecuados para cada situación.
  - Iniciar o proponer un tema.
  - Desarrollar un tema.
  - Dar por terminada una conversación.
  - Conducir la conversación hacia un tema nuevo.
  - Desviar o eludir un tema de conversación.
  - Relacionar un tema nuevo con uno viejo.
  - Saber abrir y cerrar un discurso oral.
- Conducir la interacción
  - Manifestar que se quiere intervenir (con gestos, sonidos, frases).
  - Escoger el momento adecuado para intervenir.
  - Utilizar eficazmente el turno de palabra:
    - aprovechar el tiempo para decir todo lo que se considere necesario;
    - ceñirse a las convenciones del tipo de discurso (tema, estructura, etc.);
    - marcar el inicio y el final del turno de palabra.
  - Reconocer cuando un interlocutor pide la palabra.
  - Ceder el turno de palabra a un interlocutor en el momento adecuado.

CONTENIDO DISCIPLINAR: Inglés Técnico I

5 Information technology 1

A

Information systems **collect**, **organize**, **store**, **process**, **retrieve** and **display** information in different formats (text, video, and voice). Information technology allows very fast, automated manipulation of **digital** data and their transformation from and to **analogue**.

Two basic technologies have been responsible for the development of the necessary **hardware**: **integrated circuits** and **digital communications**. Parallel advances have been made in **software**, particularly easy-to-use software products to **create**, **maintain**, **manipulate**, and **query files** and **records**. Many of these **software programs** are designed for use both by computer professionals and enthusiastic amateurs. Another important factor is the development of **computer networks** (▸▸ 6).

B

As technology develops, new *models* and *types* of computer appear. At the heart of all computers is the *hardware*. However, without *software*, computers are just dumb boxes, unable to perform any calculations or operations.

Models and types of computer

desktop • laptop • mainframe • notebook • server • terminal • workstation

Computer hardware

CPU (central processing unit) • dot matrix printer • expansion card • inkjet printer • keyboard • laser printer • monitor • mouse • RAM (random access memory) • scanner • screen • storage devices

Software

applet • application software • browser • database software • email software • graphics software • operating system • search engine • spreadsheet • word processing

C

Many words in the field of IT come from American English. So you may see the following spellings:

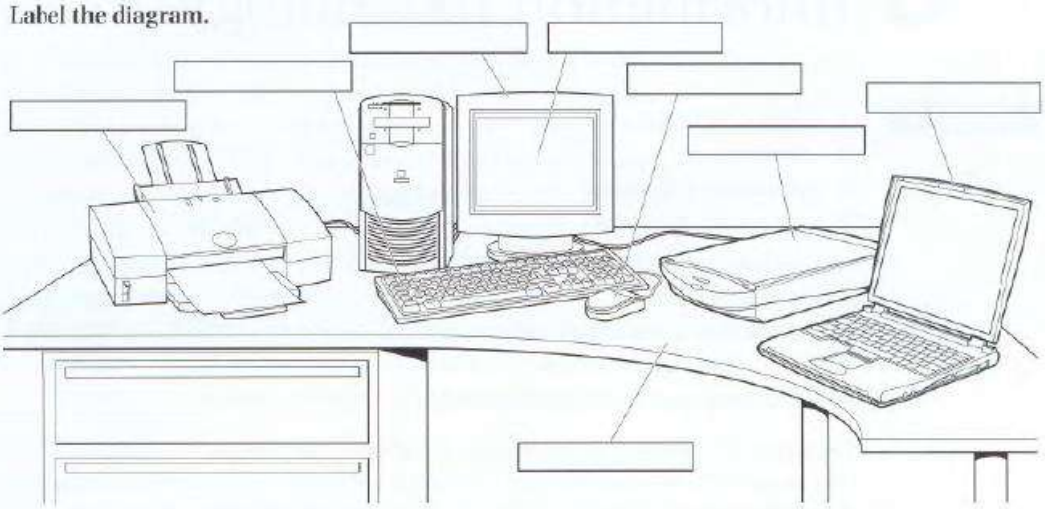
British English	American English
programme	program
analogue	analog

The area of IT is developing very quickly; and the language to describe hardware, software and applications is also evolving at a high speed. As a result new noun + noun combinations often change to single nouns:

noun + noun	single noun
lap top	laptop
note book	notebook
work station	workstation
desk top	desktop

CONTENIDO DISCIPLINAR: Inglés Técnico I

1 Label the diagram.



2 Combine one word from A and one word from B and match it with the appropriate definition in C.

A	B	C
create	products	a monitor will do this on a computer screen
central	information	this describes the format of 0 and 1 in which information is stored
software	processing unit	these enable a computer to perform word processing, to create databases, and to manipulate numerical data
display	card	when two or more components are combined and then incorporated into a single package
digital	files	to make new programs, utilities or documents
expansion	network	a group of electronic machines connected by cables or other means which can exchange information and share equipment (such as printers and disk drives)
integrated	data	the principal microchip that the computer is built around
computer	circuits	you plug this into a slot to add features such as video, sound, modem and networking

3 Complete each gap in the following text with a phrase from the table above.

1 The computer monitor will _____ _____ so you can see it on screen.

2 Information is stored on a computer as _____ _____.

3 Spreadsheet and graphic software are examples of _____ _____.

4 Digital communications and _____ _____ have allowed developments in hardware to be made.

5 In order to organise data you should _____ _____ where you can store data.

6 When several computers are linked together you have a _____ _____.

7 The part of the computer which interprets and carries out instructions is the _____ _____.

8 An _____ _____ can be inserted in your computer to give your computer extra capabilities.

CONTENIDO DISCIPLINAR: Inglés Técnico I

6 Information technology 2

A

A network includes:

- – techniques
- – physical connections
- – computer programs

used to link two or more computers.

Network users can:

- – share files, printers and other resources
- – send electronic messages
- – run programs on other computers

Each network operates according to a set of computer programs called network protocols for computers to talk to one another. Computer networks can now be interconnected efficiently through gateways. The biggest network is the World Wide Web. It consists of a large number of smaller interconnected networks called internets. These internets may connect tens, hundreds or thousands of computers. They can share information with each other, such as databases of information. The internet allows people all over the world to communicate with each other effectively and inexpensively.

B

Before a network can operate, it needs physical *connections* so that signals can be transmitted. After the network has been connected, it is ready for *operation*.

Network connections

bandwidth • baud • bits per second (bps) • optical fibre • packet • receive • signal • transmit • transmission speed • twisted pair

Network operation

configure • download • hack • hub • install • internet service provider (ISP) • local area network (LAN) • switch • transmit • upload • web page • website • wide area network (WAN) • wireless

C

A prefix comes at the beginning of a word and usually has a specific meaning, for example inter = between.

Look at the following prefixes and their use in the above IT words/phrases:

prefix	meaning of prefix	example of use
inter-	between	internet, interconnect, interactive, international
intra-	within	intranet, e.g. company intrsnet
trans-	across	transmit, transfer, transaction
co-/com-/con-	with	combine, compatible, connect, configure
up-	up (to internet)	upload
down-	down (from internet)	download, downtime, i.e. when the network is down (not working)

CONTENIDO DISCIPLINAR: Inglés Técnico I

6 Information technology 2

A

A **network** includes:

- – **techniques**
- – **physical connections**
- – computer programs

used to **link** two or more computers.

Network users can:

- – **share files**, printers and other resources
- – send **electronic messages**
- – **run** programs on other computers

Each network operates according to a set of computer programs called network **protocols** for computers to talk to one another. Computer networks can now be **interconnected** efficiently through **gateways**. The biggest network is the **World Wide Web**. It consists of a large number of smaller interconnected networks called **internets**. These internets may **connect** tens, hundreds, or thousands of computers. They can share information with each other, such as **databases** of information. The internet allows people all over the world to **communicate** with each other effectively and inexpensively.

B

Before a network can operate, it needs physical *connections* so that signals can be transmitted. After the network has been connected, it is ready for *operation*.

Network connections

bandwidth • baud • bits per second (bps) • optical fibre • packet receive • signal • transmit • transmission speed • twisted pair

Network operation

configure • download • hack • hub • install • internet service provider (ISP) local area network (LAN) • switch • transmit • upload • web page • website wide area network (WAN) • wireless

C

A prefix comes at the beginning of a word and usually has a specific meaning, for example inter = between.

Look at the following prefixes and their use in the above IT words/phrases:

prefix	meaning of prefix	example of use
inter-	between	internet, interconnect, interactive, international
intra-	within	intranet, e.g. company intranet
trans-	across	transmit, transfer, transaction
co-/com-/con-	with	combine, compatible, connect, configure
up-	up (to internet)	upload
down-	down (from internet)	download, downtime, i.e. when the network is down (not working)

CONTENIDO DISCIPLINAR: Inglés Técnico I

1 **Choose the correct word in each of the following.**

1 The speed with which a modem can process data is measured in _____.
a) bandwidth b) bits per second (bps) c) signal

2 Cables consisting of several copper wires each with a shield are known as _____ cables.
a) twisted pair b) optical fibre c) power cables

3 Computers that are connected together within one building form a _____.
a) WAN b) ISP c) LAN

4 If you transfer a file from a remote computer to your computer, you _____.
a) download b) upload c) run

5 To send out information is to _____.
a) signal b) packet c) transmit

6 A document containing information and graphics that can be accessed on the internet is _____.
a) a website b) a web page c) the World Wide Web

2 **Complete the words in the following sentences by adding the prefix *inter-, intra-, trans-, com-, con-, up- or down-*.**

1 Last month computer _____time cost the company over €10,000 in lost production.
2 The computers in the production department have now been successfully _____connected with those in the planning department.
3 Once you have completed payment details the data will be _____mitted via a secure link.
4 We cannot network these computers because the systems are not _____patible.
5 Many companies distribute internal documents on their own _____net.
6 Once the home page has been completed, we'll be ready to _____load the site.
7 Cables are being laid throughout the building as the network requires physical _____nections.
8 Using the network he was able to _____bine the data from different reports.

3 **Here is a list of instructions for someone wanting to set up a small network. Put the instructions in the correct order.**

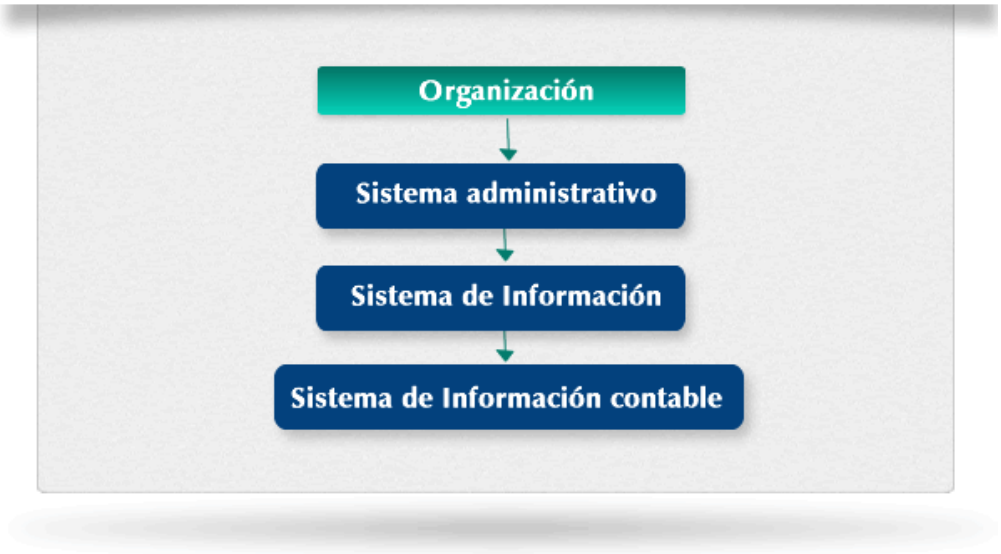
a Make wiring and layout plans for your network.
b Hook up the network cables by connecting everything to the hub.
c Check that each computer has an IP address and give it a name.
d If you're installing a small network, twisted pair will be adequate. However, in order to span greater distances and to minimize magnetic and electrical interference use fibre optic cable.
e Decide on the type of network you want to install. To enable you to transfer large amounts of data, choose Fast Ethernet (100BaseT).
f Install network adapters in the computers.
g Add an internet gateway to your network to set up a shared internet connection.
h Install driver software for the adapter driver and install client software to share printers and files.
i Check which protocols are installed and add any other protocols you require.
j Get the hardware you need: an Ethernet adapter card for each computer that doesn't have an Ethernet port, a hub if you've got more than two computers, cables and wall jacks.

CONTENIDO DISCIPLINAR: Administración

CONTENIDO DISCIPLINAR: Administración

La Contabilidad como Sistema de Información y Control

Un **Sistema de Información Contable** forma parte del Sistema de Información de las organizaciones y tiene por finalidad **reunir datos** de naturaleza contable, **procesarlos** utilizando un proceso adecuado de procesamiento, crear y mantener archivos contables y **producir información contable** bajo distintas formas, para distintos usuarios.



Dijimos que el sistema de información forma parte del sistema administrativo. Ahora completamos el concepto diciendo que el sistema contable forma parte a su vez del sistema de información.

Reunir Datos: en un Sistema de Información Contable la mayoría de los datos surgen de los comprobantes. Los comprobantes son documentos comerciales por medio de los cuales se formalizan las operaciones o transacciones. Los comprobantes tienen distintas finalidades:

- sirven de constancia de las operaciones realizadas,
- sirven como información de las operaciones efectuadas,
- sirven de base de los registros contables,
- facilitan la tarea de fiscalización y control fiscal.

Procesar Datos: el Sistema de Información Contable utiliza la técnica contable de registro de operaciones. La contabilidad ha desarrollado un procedimiento lógico, de validez y aplicación universal, que permite procesar datos y suministrar información contable para distintos usuarios: propietarios, directores, inversionistas, bancos, Estado, etc.

El proceso de registración contable comprende las siguientes instancias:

Lectura del comprobante

CONTENIDO DISCIPLINAR: Administración

Determinación de la operación realizada sobre la base del comprobante

Extracción de los datos necesarios para el registro contable

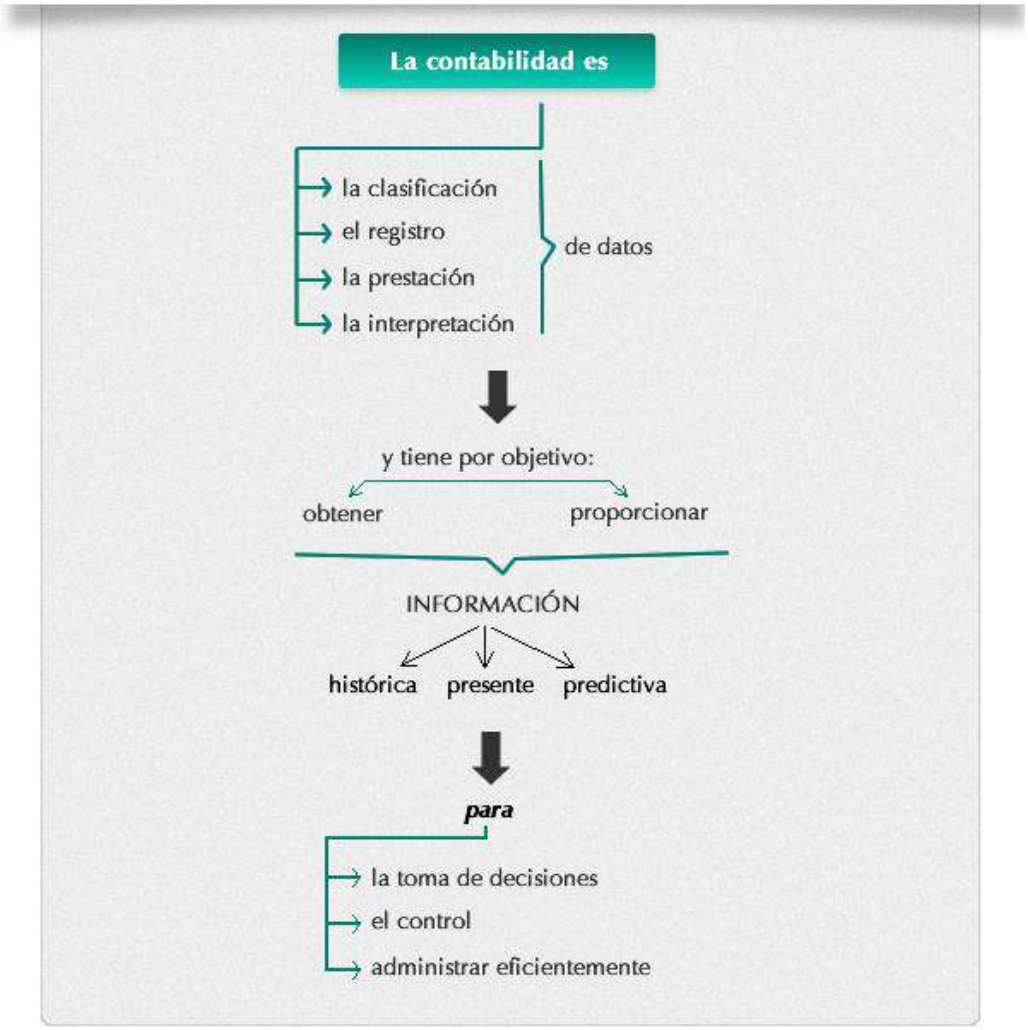
Registro contable de la operación

Producir Información Contable: es la finalidad del Sistema de Información Contable, esta información debe ser útil a los distintos usuarios.

La **Contabilidad** es un Sistema de Información que permite conocer el patrimonio y sus variaciones, controlar el desarrollo de la actividad y medir sus resultados. Brinda información sobre lo pasado, lo que está ocurriendo en este preciso momento y lo que se proyecta para el futuro.

La contabilidad es una ciencia que por medio de un **sistema de registración** (manual o computarizado), en **libros** y**fichas** con rayados especiales, permite conocer el estado del **Patrimonio** (bienes propios, derechos a cobrar y deudas) y el **Resultado** obtenido (pérdidas o ganancias) de un comercio o empresa.

La contabilidad registra en forma histórica, exacta y fiel todas las operaciones que realiza el comerciante, clasificándolas correctamente y registrándolas en forma metódica y justificada en unos libros especiales llamados "de contabilidad" y de acuerdo con las leyes comerciales, permitiendo a su vez obtener resúmenes de cifras a través de los cuales, una vez analizados,



CONTENIDO DISCIPLINAR: Administración

nos permitirá apreciar los resultados de la empresa en unos cuadros denominados "Estados Financieros" o "Estados Contables".

"Una Contabilidad bien organizada y al día, constituye la brújula que guía a la empresa, camino al éxito"

Finalidades de la Contabilidad

- Sirve de medio de información y control a sus propietarios y personal para una correcta toma de decisiones. Es la base para la toma de decisiones y ayuda a orientar el futuro de la empresa.
- Sirve de medio de información a los terceros que realicen operaciones con la empresa.
- Nos permite informarnos de lo que debemos y lo que nos deben.
- Permite controlar los gastos y las inversiones.
- Informa cuánto cuesta producir un artículo y en cuanto se lo puede vender.
- Permite conocer oportunamente cuánto estamos ganando o perdiendo.
- Con una contabilidad organizada será más fácil conseguir préstamos y asesoría.
- Es orientadora, porque nos permite conocer en un momento dado, la situación financiera (BALANCE GENERAL) y situación económica (ESTADO DE RESULTADOS) del negocio.

¿Qué necesitan los individuos y las organizaciones en el mundo laboral y empresarial?

En la sociedad actual, una gran parte de la vida se desarrolla en instituciones o empresas, o en estrecha relación con ellas. Todas las organizaciones, con o sin fines de lucro, públicas o privadas, constituyen parte sustancial del ambiente social en el que las personas interactúan. Todos participamos de actividades económicas, ya sea como: consumidores, productores de bienes o servicios, o ambos a la vez. Todos somos actores en la vida económica, aunque de distintas formas:

- En un puesto de trabajo en relación de dependencia dentro de una organización (obrero/empleado).

- Haciendo trabajos por cuenta propia para otras personas o empresas (trabajador independiente o autónomo).

CONTENIDO DISCIPLINAR: Administración

- En un negocio propio que administramos (empresario).
- Mediante otras actividades también de carácter económico, como: alquilar bienes muebles o inmuebles de nuestra propiedad, invertir en acciones de empresas o en títulos del Estado, etc.

En todos los casos y modalidades, las personas se relacionan con organizaciones y éstas, a su vez, no actuan en el vacío, sino dentro de determinados contextos.

Contexto Internacional

Contexto Nacional

Contexto Local

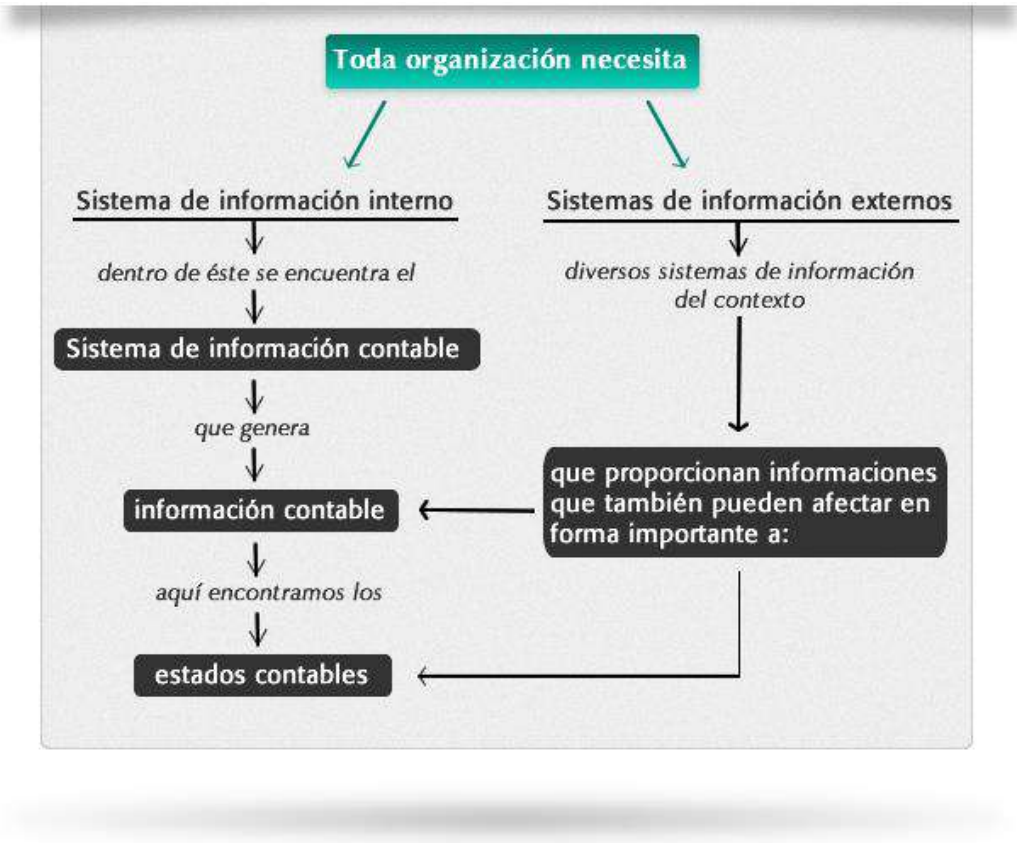
Individuo - Organización

En los momentos actuales, son cada vez más imperiosas las necesidades de **flexibilidad, rapidez de reacción a los cambios, saber detectar a tiempo las desviaciones**. Para estos tres aspectos, se necesita **INFORMACIÓN**.

Los Sistemas de Información (de todo tipo) deben estar diseñados de manera que permitan: percibir los CAMBIOS que se vayan produciendo en el ENTORNO y obtener INFORMACIÓN ÚTIL para el logro de metas y objetivos de la organización

¿Qué Sistemas de información Necesitan?

CONTENIDO DISCIPLINAR: Administración



¿Para qué se necesita Información Contable?

Entre las informaciones que genera un **Sistema Contable** se encuentran los estados contables que deben proporcionar información de calidad, principalmente cuantitativa para:

- tomar decisiones económico-financieras,
- controlar su gestión; comparando lo real con lo planeado y el procesamiento de la información.

¿Cómo necesita que sea la información?

Toda entidad necesita información contable confiable para que sea útil. Es necesario también que esa información reuna determinados requisitos generales que le den garantía de calidad que es posible mediante aplicación de **normas contables.**

CONTENIDO DISCIPLINAR: Administración

Normas contables generales y particulares

Sobre valuación y medición

Sobre exposición

De la información contable

¿Por qué se necesita información Contable de Calidad?

La Información Contable debe reflejar la situación de la empresa y su relación con el entorno, expresando una conformidad razonable con la realidad, esto significa que:

- debe expresar los acontecimientos tal como son;
- considerar elementos y valores que tienen efectiva vigencia;
- debe dar prioridad a su naturaleza económica por sobre otras consideraciones;
- debe permitir su comprobación mediante demostraciones que la acrediten y confirmen;
- debe buscar un acercamiento a la exactitud en la medida de lo posible;

debe prepararse conforme a criterios normas y reglas que le otorguen carácter de creible (para que tenga confiabilidad)

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo

INTRODUCCIÓN

El propósito de este primer encuentro es presentar la materia de El Mundo del Trabajo y su rol en el marco de la carrera. Puede constituir un aporte contar con la participación de un/a tutor/a o referente de los Programas PROG.R.ES.AR. y Jóvenes con Más y Mejor Trabajo durante la presentación. Durante este encuentro se espera que el equipo docente y el grupo de jóvenes puedan conocerse entre sí y construir algunos acuerdos básicos para el funcionamiento del desarrollo de la materia.

ACTIVIDADES

ACTIVIDAD 1: QUIÉNES SOMOS

OBJETIVOS

- Promover el conocimiento, generar un clima de confianza y de integración grupal entre los y las participantes.
- Presentar los propósitos de programas como PROG.R.ES.AR. y del PJMyMT.
- Brindar información específica respecto de los Programas antes mencionados.

RECURSOS

1. **Herramientas conceptuales:**
   - Ficha 1 para el/la docente: "Descripción del PROG.R.ES.AR y del Programa Jóvenes con Más y Mejor Trabajo"
2. **RECURSOS MATERIALES Y ÚTILES:**
   - Rotafolios, pizarra o papeles afiche
   - Tizas, fibras
   - Almohadón o pelota

DESARROLLO

A. Comience presentándose y dando la bienvenida a los y las alumnos del grupo. Explique que la Materia El Mundo del Trabajo es una actividad que se realiza en el marco del desarrollo de la carrera.
   - Escriba los nombres de los Programas y el nombre de la Materia en la pizarra o afiche para desarrollar "PROG.R.ES.AR y del Programa Jóvenes con Más y Mejor Trabajo", posteriormente, con más detalle en qué consisten.

B. Explique el objetivo general del curso. En el mismo se intentará que cada participante reflexione sobre su propio futuro laboral e identifique las herramientas necesarias que requiere para mejorar sus posibilidades de obtener el empleo deseado.

C. Presentación de cada participante. Sentados/as en círculo, quien coordina el curso dirá su nombre, luego respetando el orden de la ronda, cada uno/a dirá su nombre y repetirá el de los/las participantes que lo antecedieron hasta finalizar el círculo.

D. Terminada esta primera presentación, invite a que cada participante pueda compartir otros aspectos personales y alguna expectativa en relación al curso, en particular y a sus proyectos futuros en general:
   - Edad
   - Con quién/es vive
   - Qué hace cotidianamente
   - Qué le gusta hacer
   - Si alguna vez trabajó y en qué
   - Qué expectativas o proyectos tiene para su futuro. Si le interesa estudiar (¿qué?), si quiere trabajar, si no tiene claro qué desea hacer.

   Para llevar a cabo esta actividad utilice un almohadón o una pelota. Preséntese según la guía propuesta; luego nombre a un/a compañero/a al azar y arrójele el objeto. Quien lo recibe deberá ampliar su presentación y al finalizar repetir la acción con otro/a compañero/a, hasta terminar la rueda.

E. Siguiendo la ronda, invite a mencionar una expectativa en relación con el Curso de Introducción al Trabajo. Regístrelas en un papel afiche y aclare cuáles pueden ser satisfechas y cuáles no.

F. A continuación proponga al grupo realizar una "lluvia de ideas" en la que cada uno/a comparta lo que sabe o cree saber acerca de las características y objetivos de los Programas dentro de nuestro país como

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo

PROG.R.ES.AR. y Jóvenes con Más y Mejor Trabajo. Registre en un afiche en dos columnas las ideas correctas y las incorrectas respecto de las prestaciones que los programas pueden brindar.

G. Desarrolle y complete la información enumerando objetivos, requisitos y acciones que proponen los programas. Se sugiere procurar que los/las tutores/as y referentes de los programas participen del encuentro, a los efectos de que aclaren dudas acerca de distintos aspectos de los mismos.

ACTIVIDAD 2: LA MATERIA EL MUNDO DEL TRABAJO

OBJETIVOS

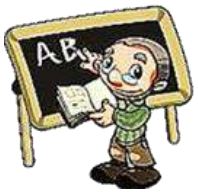
Presentar la Materia El Mundo del Trabajo.

RECURSOS

1. **HERRAMIENTAS CONCEPTUALES:**
   - Para el/la docente: "Proyecto Ocupacional"
2. **RECURSOS DIDÁCTICOS**:
   - Hojas con instrucciones y plano para el juego "UN ASADO CON AMIGOS"
3. **RECURSOS MATERIALES Y ÚTILES:**
   - Pizarra o papel afiche
   - Tizas o fibras
   - Cinta adhesiva

DESARROLLO

A. Invite a los/as jóvenes a conformar pequeños grupos para participar del juego "UN ASADO CON AMIGOS". Distribuya las hojas con el plano y las instrucciones. Explique que deberán leer atentamente las tareas indicadas y proponer un orden en el cual las realizarían.



UN ASADO CON AMIGOS

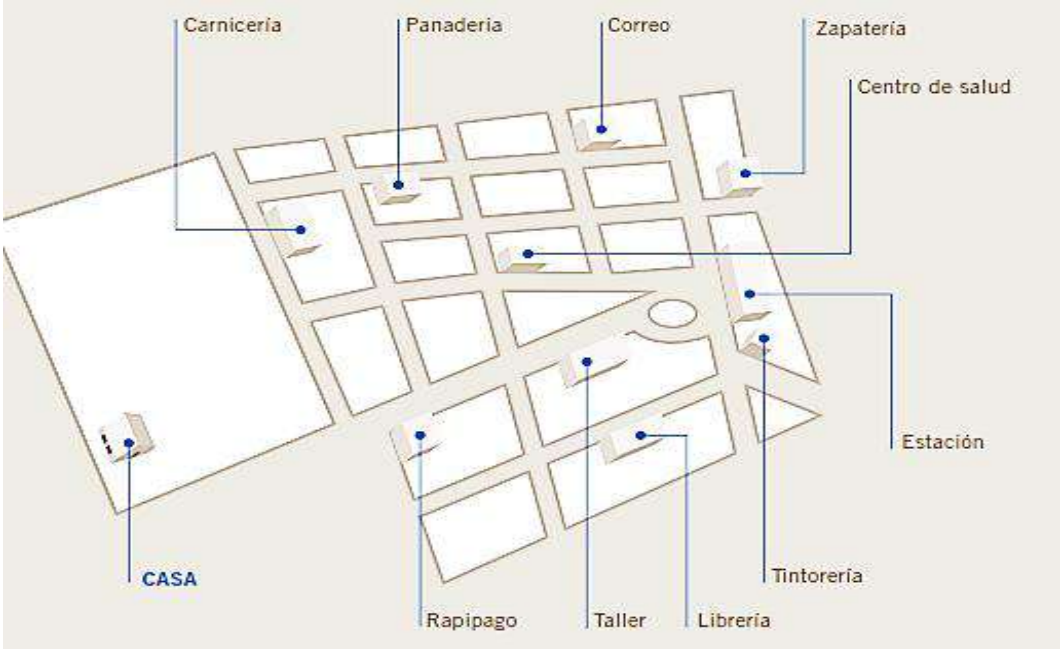
Felipe, que hace trabajos de cadete en una oficina del barrio, tiene programado para hoy comer un asado con amigos que hace tiempo no ve. Debe organizar todos los aspectos vinculados con el asado y además resolver antes algunas cuestiones para poder disfrutar con tranquilidad de las visitas.

*Debe salir de su casa a las **9:15**, hacer una serie de tareas y regresar con sus amigos a las **13:00 horas**. Se encontrará con ellos a las 12:00 en la estación de tren. Las tareas son las siguientes:*

1. *Llevar unos zapatos al zapatero.*
2. *Retirar una computadora de la oficina del taller de reparaciones.*
3. *Llevar un abrigo a la tintorería.*
4. *Enviar una encomienda de 10 kg. por correo.*
5. *Sacar un turno médico.*
6. *Pagar los impuestos de la oficina en el Rapipago.*
7. *Comprar el pan para el asado.*
8. *Esperar a sus amigos que llegan a la estación a las 12:00.*
9. *Retirar un libro encargado en la librería.*
10. *Comprar la carne para el asado.*

Para organizar su jornada deberá tener en cuenta que: para recorrer el camino de su casa a la estación tarda 30 minutos; el centro de salud termina de dar turnos a las 10:00; los comercios y el correo cierran a las 12:30 y el Rapipago abre a las 11:00. El recorrido debe hacerse a pie

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo



B. Luego del tiempo necesario, coordine una puesta en común en la cual cada grupo explique cuál sería el orden de las tareas y justifique por qué. Vaya registrando en la pizarra o en el afiche las alternativas propuestas.
C. Solicite al grupo de participantes que agrupen las actividades en tres rubros: las vinculadas a la preparación del asado; las personales y las laborales. Explique que para el logro del objetivo –hacer el asado–, debe dar cumplimiento a otras actividades que forman parte de sus obligaciones o responsabilidades.
D. Realice un cierre resaltando la variedad de estrategias, actitudes, habilidades, que cada grupo fue poniendo en juego para resolver las instrucciones. Señale que para resolver el problema seguramente han considerado cuestiones tales como:
   • La velocidad para caminar
   • La posibilidad de cargar más o menos peso
   • Los factores externos, como los horarios de los comercios y las distancias
   • La inclusión o no del pedido de ayuda para repartir tareas
E. Proponga entonces una semejanza entre las actividades que realizaron en el juego y la construcción de un proyecto. La meta del juego era poder compartir un asado con amigos: para ello no sólo debía comprarse lo necesario y encontrarlos en la estación, sino además resolver otras cuestiones.
   Explique que, de manera semejante al juego, cuando una persona se propone alcanzar determinada meta, tiene que:
   • Definir la meta (¿Qué?)
   • Considerar actividades y tareas posibles para lograrla (¿Cómo?)
   • Estimar los recursos necesarios y los disponibles (¿Con qué?)
   • Calcular el tiempo requerido (¿Cuándo/cuánto?)
   • Considerar las dificultades o cuestiones externas que no dependen de uno (¿Qué problemas hay que enfrentar?)
   A medida que avanza en su exposición vaya registrando en un papel afiche las preguntas consignadas más arriba, que permiten caracterizar un proyecto.

HERRAMIENTA CONCEPTUAL

PROYECTO OCUPACIONAL

Entendemos por proyecto ocupacional al conjunto de cursos de acción que una persona define, planifica, ejecuta, revisa y replanifica con vistas a lograr una inserción productiva, o a mejorar su situación en el empleo (FORMUJER).

La metodología del proyecto ocupacional se apoya en una práctica que todas las personas realizan en su vida cotidiana, en la práctica de proyectar, de definir qué camino seguir para resolver un problema, para potenciar una oportunidad, para alcanzar un

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo

determinado resultado. Un proyecto se origina en el análisis de una situación que se quiere modificar o enfrentar, y a partir del mismo pueden surgir nuevas alternativas y objetivos a lograr, junto a diversos caminos para llegar a ellos.

Un proyecto ocupacional tiene la misma lógica. Para construirlo, las personas deben comenzar por identificar su situación actual o punto de partida, así como ubicar cuál es el campo laboral en el que aspiran insertarse o "meta". Esto significa, por una parte reconocer los conocimientos, habilidades, actitudes que cada persona ha adquirido en sus trayectorias de vida y formativas, y seleccionar aquellos que puedan tener un valor en el mercado de trabajo. Y por otra, contrastar sus conocimientos y habilidades con las características y calificaciones demandadas en el campo laboral elegido. Este contraste les permite obtener una "fotografía" de su situación actual, en la que pueden ver "lo que tienen" y "lo que les falta" para alcanzar su meta laboral. Este ejercicio les brindará la posibilidad de definir sus objetivos en materia de formación y/o de inserción o mejora de su situación laboral y de trazar un plan de acción para alcanzarlos.

En el proceso de construcción del proyecto ocupacional las personas fortalecen su empleabilidad, ya que desarrollan capacidades para tomar decisiones en lo laboral con mayor autonomía luego de haber reflexionado sobre su propia condición y sobre el contexto laboral. Además, desarrollan capacidades para planificar, gestionar un uso eficiente de los recursos disponibles, comunicar sus capacidades y resolver problemas, entre otras.

¿POR QUÉ ESTE ENFOQUE FORTALECE LA EMPLEABILIDAD E INTEGRA LA PERSPECTIVA DE GÉNERO?

- *Porque se centra en lo que las personas tienen y lo pone en valor.*
- *Porque lo que les falta no se connota como estigma, sino como generador de nuevas oportunidades.*
- *Porque focaliza en el sujeto en su contexto, haciendo visibles los condicionamientos de género, condición social, etnia, edad, entre otros.*
- *Porque apuesta a la proyección personal y a la posibilidad de cambios. Aun en contextos de incertidumbre y restricciones hay márgenes para la acción.*
- *Porque permite desarrollar competencias laborales transversales.*

ETAPAS EN LA CONSTRUCCIÓN DE UN PROYECTO OCUPACIONAL

Etapa 1: Autodiagnóstico y análisis del contexto: *Es el punto de partida, donde se toman como referencia nuestros saberes previos, experiencias laborales y extralaborales para contrastarlo con los requerimientos de un puesto de trabajo.*

Etapa 2: La definición de los objetivos: *Aquí se trata de definir qué se quiere lograr, en cuánto tiempo y con qué recursos.*

Etapa 3: El plan de actividades: *En esta etapa se deberá trazar un plan de actividades para lograr los objetivos, calcular los tiempos que llevará cada una y los recursos. Si el objetivo es lograr la inserción laboral, el plan de actividades deberá organizar el proceso de búsqueda de empleo. Además, si el objetivo incluye la realización de actividades formativas se deberá prever la elección del curso, la ubicación, los horarios, etc.*

Etapa 4: Ejecución y evaluación: *Una vez transitadas las etapas de diseño del proyecto ocupacional, llega el momento de probarlo, de ponerlo en marcha en el contexto real.*

F. Agregue que, de manera similar a como se resolvió el juego, se trata de pensar cuál es la meta en relación al futuro laboral de cada joven del grupo; cuáles son sus capacidades, sus habilidades; qué necesitan o qué les falta para lograr la meta y qué pasos es necesario dar para ello. También en este caso tendrán que considerar que pueden existir factores personales que favorezcan o dificulten alcanzar la meta.
G. Destaque que en este proceso podremos contar con la ayuda de compañeros/as que aporten su mirada, ideas, ayuden a pensar, a descubrir posibilidades... Y además, con la presencia del docente que actúen como guías y faciliten la información necesaria. Esta modalidad de trabajo permitirá que cada uno/a pueda construir su propio proyecto.
H. Cierre el bloque ofreciendo información sobre el Curso de Introducción al Trabajo:
   - El Proyecto Formativo Ocupacional se construirá durante cuatro meses en el marco del curso.
   - Recuerde cuáles son los módulos que integran el curso; sus objetivos; sus características; su duración; etc.
   - Destaque la función del equipo de coordinación de los cursos y la importancia del compromiso y la participación activa del grupo de jóvenes en este curso.
I. Por último, abra una ronda de preguntas y aclare las dudas que se planteen.

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo

ACTIVIDAD 3: CÓMO SERÁN LOS ENCUENTROS: NUESTROS COMPROMISOS

OBJETIVOS

• Establecer acuerdos y compromisos para el trabajo grupal.

RECURSOS

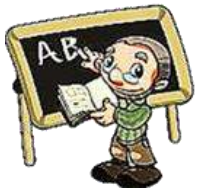
1. **RECURSOS DIDÁCTICOS**:
   - etas con frases escritas
   - Ficha de registro y evaluación individual ("Ficha de datos personales")
2. **RECURSOS MATERIALES Y ÚTILES:**
   - Pizarra o papel afiche
   - Tizas o fibras

DESARROLLO

A. Introduzca esta actividad mencionando que, a lo largo del desarrollo de la Materia, los encuentros serán el ámbito en el que cada participante hará su proceso personal para definir su propio proyecto. Por eso es importante acordar algunas pautas de funcionamiento grupal que permitan un clima amigable para el trabajo.

B. Proponga a los/las participantes formar grupos de tres o cuatro integrantes y pídales que lean del cuadernillo las frases que previamente haya asignado para cada grupo, invitándolos a dialogar sobre sus opiniones acerca del contenido de cada una de ellas:

FRASES



- *No voy a decir lo que pienso, ¿a quién le puede interesar?*
- *Algunas personas todo el tiempo dicen cosas fuera de lugar.*
- *Todos podemos cambiar.*
- *Aunque seamos diferentes, todos tenemos algo para aportar.*
- *Hay que animar a los más tímidos, para que también se expresen.*
- *Yo mejor no hablo, a ver si después me discriminan…*
- *Con todo lo que tengo que hacer… venir a hablar de estas pavadas…*
- *Aprovechan el espacio del taller para chusmear lo que pasa en el barrio.*
- *No estamos de acuerdo con el tallerista, pero mejor no se lo decimos, a ver si nos echa.*
- *Si surge alguna dificultad podemos ver de resolverlo entre todos; también se lo podemos plantear al profesor…*
- *¡Qué costumbre de hablar todos a la vez! ¿Por qué no se callan y dejan que el profesor diga todo?*
- *¡Alguien del grupo dijo algo que a mí nunca se me había ocurrido! ¡Y está bueno!*
- *A uno de mis compañeros no lo soporto. Dice cosas que me duelen.*
- *No tengo ganas de hacer nada. Que trabajen los demás.*
- *Tengo un problema personal con un compañero. Pero prefiero hablarlo cuando esté tranquilo y no discutir delante de todos…*
- *Lo que me pasa es cosa mía, no tiene nada que ver con la clase. Que nadie se meta en mi vida.*
- *Me parece que uno de mis compañeros no está bien… Quizás en algún corte le pregunto si necesita algo.*
- *No entiendo nada de lo que dicen. ¿Qué hago?*
- *¡Si decimos las cosas de buena manera y con respeto podemos hablar de muchas cosas!*
- *Es mejor que hablen los que saben. Los demás se callan la boca.*
- *Ojalá haya varios equipos de mate, para que todos podamos tomar durante el encuentro.*
- *Hay algunos que no pueden despegarse del celular ni un momento… Hasta están con los auriculares puestos todo el tiempo…*

C. Después de unos minutos de discusión en los pequeños grupos, coordine una puesta en común rescatando las conclusiones alcanzadas en cada uno de ellos. Luego vaya registrando en un afiche "Qué queremos" y "Qué NO queremos" para este grupo.

D. Pregunte qué otras cuestiones son importantes para el buen funcionamiento del grupo, por ejemplo:
   - Respetar horarios y tiempos (de llegada y de salida, de refrigerios)
   - Comunicar si existiera alguna dificultad para asistir
   - Establecer pautas en relación a:
     - El uso de celulares
     - Lugares y momentos para tomar mate y/o fumar

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo

- El cuidado de las condiciones del aula
- Otras cuestiones

Si estas ideas no aparecieran espontáneamente, propóngalas usted destacando su importancia y señalando que cuestiones como la puntualidad, la comunicación, el compromiso con la tarea, son competencias que luego tendrán que poner en juego en el mundo laboral y éste puede ser un escenario para ejercitarlas.

E. Para cerrar esta actividad, confeccione de manera conjunta con el grupo un afiche titulado "Nuestros compromisos" que incluya el listado de aquellos acuerdos que el grupo asume como propios para asegurar el buen funcionamiento.

F. Para cerrar este encuentro proponga realizar una breve evaluación, en la que quienes lo deseen mencionen brevemente aspectos positivos y negativos del mismo o simplemente cuenten cómo se sintieron.

G. Finalmente, solicite que cada uno/a complete la ficha de registro individual. Conserve estas fichas como herramientas para el acompañamiento a lo largo del desarrollo de la materia.

FICHA DE DATOS PERSONALES

NOMBRE Y APELLIDO: _____

FECHA DE NACIMIENTO:_____

CILIO: _____

BARRIO / LOCALIDAD: _____

C.P.: _____

TELÉFONO DE LÍNEA:_____ _____

TELÉFONO CELULAR:_____

CORREO ELECTRÓNICO: _____

FACEBOOK: _____

Agradezca la participación y recuerde los horarios del próximo encuentro.

ACTIVIDAD 4: TRABAJADORAS Y TRABAJADORES

OBJETIVOS

• Establecer y reconocer las diferentes actividades que muestran los diferentes dibujos.

RECURSOS

1. **RECURSOS DIDÁCTICOS**:

etas con dibujos de diferentes tipos de actividades

• Ficha de registro de las diferentes actividades

2. **RECURSOS MATERIALES Y ÚTILES:**

• Pizarra, papel afiche, fotos

• Tizas o fibras

• Proyector

DESARROLLO

A. Los dibujos muestran a hombres y mujeres realizando diferentes actividades..

B. Aunque realiza tareas diferentes en distintos lugares y usan distintas herramientas, ¿Puede decirse que todas estas personas están trabajando? ¿Por qué?, llene la cartilla correspondiente
En este momento, millones de personas trabajan en el campo, en los pueblos y en las pequeñas y grande ciudades. Cada una de ellas tiene una historia que lo distingue de todas los demás. A pesar de todas las diferencias que puede haber entre estas personas, todos ellos, hombres y mujeres son trabajadores.

C. Pero el trabajo no solo es una actividad que comparte millones de hombres y mujeres. Tambien es una actividad que permite *identificar* a las personas. Por ejemplo, seguramente usted habrá vivido parecida a la siguiente.

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo

Así como el nombre y el apellido distinguen a una persona de todas las demás, el trabajo de una persona realiza también es importante para identificarla

Antiguamente, en ciertos pueblos, las personas recibían como apellido el oficio que realizaban, actualmente, algunos apellidos conservan este origen. Por ejemplo herrera o Molina.

ACTIVIDAD 5: IDENTIFICANDO NUESTRO APELLIDO CON UN OFICIO

OBJETIVOS

• Establecer y reconocer las diferentes tipos de apellidos relacionados con un oficio o tipo de trabajo.

RECURSOS

1. **RECURSOS DIDÁCTICOS**:

• Tarjetas
• Ficha de registro de las diferentes actividades

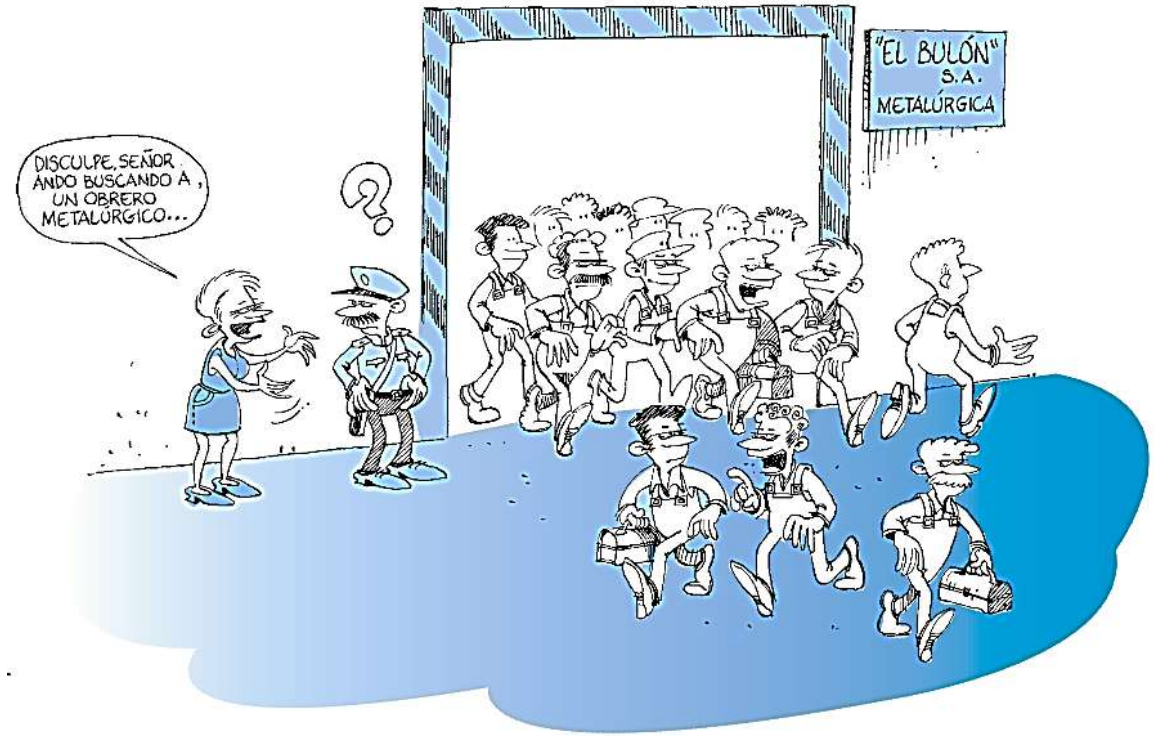
2. **RECURSOS MATERIALES Y ÚTILES:**

• Pizarra, papel afiche, fotos
• Tizas o fibras
• Proyector

DESARROLLO

A. Mencionar por los menos dos apellidos que contengan el nombre de un oficio o tipo de trabajo y sea diferente al resto de sus compañeros.
B. Hace mucho tiempo las personas se solían identificar por el trabajo y oficio que desempeñaban. En aquel entonces, muchas personas tenían un trabajo fijo para toda la vida que inclusive pasaba de generación en generación,
Actualmente, el trabajo u oficio no en todos los casos puede servir para identificar a una persona por ejemplo.

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo



Tal vez el interior de una familia o de un grupo de amigo, o en el ámbito de un pequeño pueblo, aun hoy la gente use el trabajo que desempeña una persona para diferenciarla de las demás. Por Ejemplo una obra de teatro de un famoso escritor Uruguayo *Florencio Sánchez*, se titula "*M'hijo el Doctor*"

C. Pero en un Pueblo grande en o una ciudad, el trabajo u oficio de una persona no es suficiente par identificarla. En un pueblo grande o en una ciudad, hay una gran cantidad de personas que realizan tareas similares. En una gran ciudad, hay miles de bancarios, miles de enfermeros, miles de obreros de la construcción.
Sin embrago, aún hoy, la pregunta "*¿en qué trabajás?*" es muy frecuente en el inicio de una relación.

ón:

ACTIVIDAD 6: CONOCIENDO A LAS PERSONAS Y QUE TRABAJO U OFICIO TIENE

OBJETIVOS

• Establecer y reconocer a una persona y ver que trabajo u oficio tiene o desarrolla

RECURSOS

1. **RECURSOS DIDÁCTICOS**:

• Tarjetas
• Ficha de registro de las diferentes actividades

2. **RECURSOS MATERIALES Y ÚTILES:**

• Pizarra, papel afiche, fotos
• Tizas o fibras
• Proyector

DESARROLLO

CONTENIDO DISCIPLINAR: UDI I: Introducción Al Mundo del Trabajo

A. De todos modos, cuando tratamos de conocer a una persona, seguramente nos hacemos una serie de preguntas. Además de su nombre y apellido, además de su edad y el lugar donde vive, es muy probable que usted le interese saber en que trabaja esa persona o qué tipo de oficio hace.
B. Por qué, para conocer a una persona, es importante saber en que trabaja, o que tipo de oficio hace.

_____

_____

_____

D. En nuestro país, millones de personas trabajan diariamente. Algunas de ellas tienen la misma ocupación durante toda la vida. Otras cambian varias veces de empleo. Hay personas que tienen una única ocupación. Otras trabajan en dos o tres empleo.

También es cierto que hay muchas personas que actualmente no tienen trabajo. Algunas de estas personas quedaron sin empleo. Otras buscan conseguir su primera ocupación.

____

CONTENIDO DISCIPLINAR: MATEMÀTICA

UNIDAD 1:

CONJUNTO NUMERICOS

- Números Naturales y Enteros. Propiedades
- Números Racionales. Propiedades.
- Números Irracionales. Propiedades. Notación científica
- Números Reales. Estructura algebraica
- Números complejos. Estructura algebraica

Símbolos matemáticos de uso frecuente

=	igual a	∧	y
≠	no es igual a	∨	o, en sentido inclusivo
≅	aproximado a	⊻	o, en sentido exclusivo
<	menor que	⇒	implica (condición necesaria)
≮	no es menor que	⇔	Implica doblemente(condición necesaria y suficiente)
>	mayor que	∴	Por lo tanto ; en consecuencia
≯	no es mayor que	/	Tal que
≤	menor o igual que	∃	Existe
≥	mayor o igual que	∀	Para todo
±	mas o menos	∈	Pertenece
∞	Infinito	⊆	Incluido en
∝	proporcional a	⊂	Incluido estrictamente en
//	paralelo a	⊇	Incluye a
⊥	perpendicular a	⊃	Incluye estrictamente a
∡	ángulo	∪	Unión o reunión
⊾	ángulo recto	∩	Intersección

Algunas letras del alfabeto griego

α alfa	β beta	γ gamma	δ delta
ε épsilon	λ lambda	μ mu	ρ rho
π pi	σ sigma	ψ psi	ω omega

CONTENIDO DISCIPLINAR: MATEMÁTICA

CONJUNTOS NUMERICOS

Introducción

Un número es una idea que expresa una cantidad, ya sea por medio de una palabra o de un símbolo. El símbolo de un número recibe el nombre de numeral.
Pensamos en números cuando contamos personas, vemos la hora, medimos la temperatura, comparamos velocidades, pesamos cuerpos, etc…
A lo largo de la historia cada civilización adoptó un sistema de numeración propio. En la actualidad aún se usa, el sistema de numeración romana, que se desarrollo en la antigua Roma y se utilizó en todo su imperio. Era un sistema de numeración no posicional en el que se usan letras mayúsculas como símbolos para representar a los números: I, V, X, L, C , D , M
El sistema universalmente aceptado actualmente (excepto algunas culturas) es el Sistema de Numeración Decimal.
Es un sistema de numeración en el que las cantidades se representan utilizando como base el número diez, por lo que se compone de las cifras cero (0); uno(1): dos (2); tres (3); cuatro (4); cinco (5); seis (6); siete (7); ocho (8) y nueve (9). Este conjunto de símbolos se denomina números árabes.

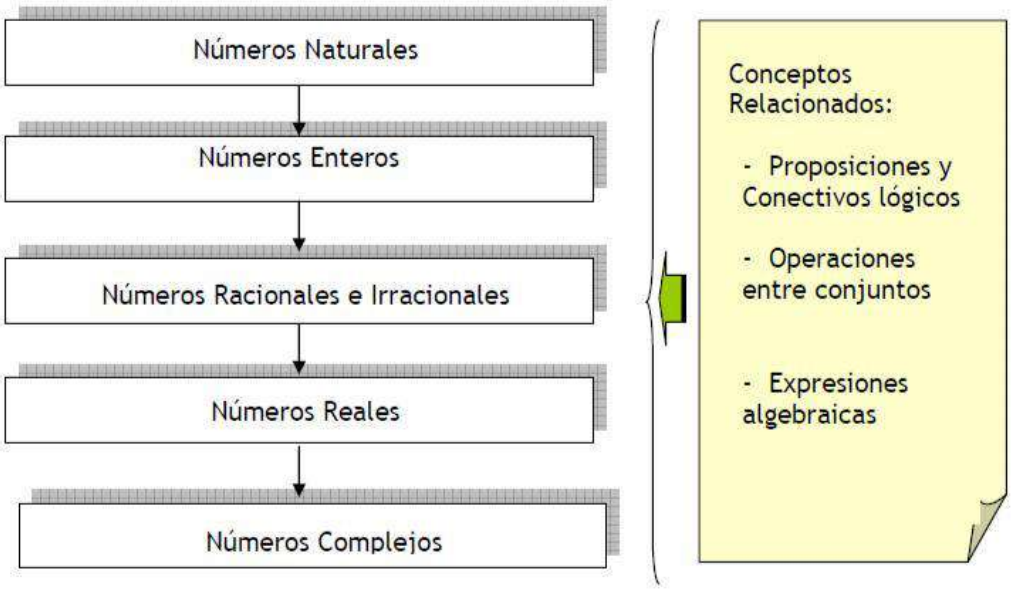
Objetivos

- Definir a los conjuntos numéricos
- Distinguir entre racional e irracional, entre real y complejo
- Recordar la aritmética de los números reales y complejos
- Adquirir habilidad en la resolución de situaciones problemática

Conceptos previos

- Conceptos básicos de lógica proposicional.
- Teoría de Conjuntos

Los números se agrupan en conjuntos o estructuras diversas; cada una contiene a la anterior y es más completa y con mayores posibilidades en sus operaciones.
Están representadas en el siguiente mapa conceptual



CONTENIDO DISCIPLINAR: MATEMÁTICA

Definición

Los números **Naturales** son los números que usamos para contar u ordenar los elementos de un conjunto no vacio
Simbólicamente: $N = \{1, 2, 3, 4, 5, ...n, n+1, ...\}$

Operaciones

La suma y el producto de números naturales son siempre naturales. En cambio la diferencia no siempre es otro natural. Simbólicamente:

Si a €*N* y b € *N*, entonces a + b € *N* (a y b se llaman términos o sumandos)
Si a €*N* y b € *N*, entonces a . b € *N* (a y b se llaman factores)

Ejemplos:

1) $3 + 7 = 10 \in N$
2) Si $n \in N$, entonces $n + 1 \in N$
3) $3 . 7 = 7 + 7 + 7 = 21 \in N$
4) Si $n \in N$, entonces $n.(n+1) \in N$
5) $3 - 3 \notin N$
6) $3 - 7 \notin N$

NUMEROS ENTEROS

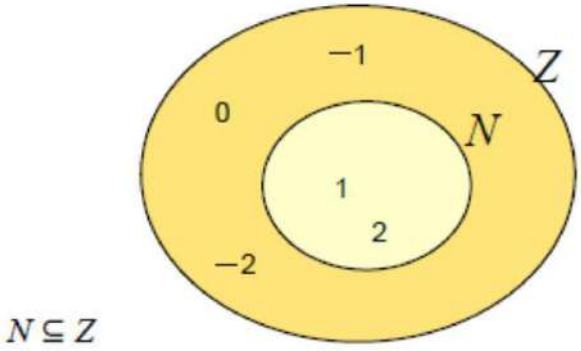
Para dar solución al problema que se presenta al restar números naturales donde el minuendo es igual o menor al sustraendo, se crearon otros números que amplia al conjunto de números naturales.
Se agregan el número cero y los números opuestos a los naturales De ese modo 3 – 3 = 0 y 3 – 7 = -4

Definición

El conjunto de los números **Enteros** está formado por la unión de los naturales, el cero y los opuestos de los naturales
Simbólicamente se expresan Z= {..... -3, -2, -1, 0, 1, 2, 3, ...}

Los números enteros permiten contar nuevos tipos de cantidades (como los saldos acreedores o deudores) y ordenar por encima o por debajo de un cierto elemento de referencia (las alturas sobre o bajo el nivel del mar o temperaturas superiores o inferiores a 0 grados, los pisos de un edificio por encima o por debajo de la planta baja, etc…).

En un gráfico de conjuntos se aprecia claramente que



$N \subseteq Z$

Se representa a los números enteros en una recta graduada, donde se elige un punto arbitrario para representar al 0 (al cual le llamaremos origen) y se adopta un segmento como unidad y la convención de que para la derecha estarán los números enteros positivos (naturales) y para la izquierda estarán los enteros negativos (opuestos de los naturales).

Operaciones en Z

La **suma** y el **producto** de enteros es siempre otro entero.

Ejemplos:

3 + 7 = 10	3 . 7 = 21
3+(-7) = -4	3.(-7) = -21
(-3) + 7 = 4	(-3).7=-21
(-3)+(-7) = -10	(-3).(-7) = 21

La **diferencia** a – b es considerada como la suma del minuendo más el opuesto del sustraendo a – b = a + (-b) donde a es el minuendo y b es el sustraendo

Ejemplos:

3 - 7 = 3 + (-7) = -4	(-3)-7=(-3)+(-7)=-10
3 - (-7) = 3 + 7 = 10	(-3) - (-7)= (-3) + 7 = 4 3 + 7 = 10

La **división** entre los enteros **a** y **b**, con b≠ 0, arroja como resultados dos números enteros llamados cociente (**q**) y resto)
A **a**se le dice dividendo y a **b** se le dice divisor.

El cociente indica las veces que el divisor está contenido en el dividendo, pudiendo quedar un resto positivo o nulo. Esto se expresa con la siguiente igualdad

$$a = b.q + r \text{ con } 0 \leq r < |b|$$

Cuando se reparte una cantidad a de elementos en b porciones, cada porción contiene q elementos pudiendo quedar r elementos sobrantes

Ejemplos:

1) 4 está contenido en 7 una vez quedando un resto de 3. Esto es 7 = 4 . 1 + 3

2) Si se quiere repartir una deuda de $45 en 8 personas, a cada una le corresponderá pagar $6 quedando un dinero a favor de $3. Esto se expresa formalmente diciendo que la división de -45:8 arroja un cociente -6 y resto 3 pues -45 = 8.(-6)+3

Caso particular: Si r = 0, entonces **a = b.q**

Se dice que la división es **exacta**, que "a es **múltiplo** de b", que "a es **divisible** por b", que "b es **factor** de a" o que "b es **divide a** a"

Ejemplos:

1) -16 es múltiplo de 4
2) 6 es factor de -24
3) -7 es divisor de -14
4) 1 y -1 son divisores de n , $n \in Z$
5) n es divisor de n , $n \in Z$, $n \neq 0$
6) 25 es múltiplo de 5
7) 8 tiene cuatro divisores positivos: 1 , 2 , 4 y 8
8) 8 tiene infinitos múltiplos positivos: { 8 , 16 , 24 , 32 , ...}

La división por 0 no está definida.
Ejemplos: 2: 0 y 0: 0 no existen!!!!!

La operación de potenciación se define como un producto particular:

Sean $a \in Z$, $n \in N$,se define la **potencia enésima de a** como el número a^n que es el resultado de multiplicar a por si misma n veces

$a^n = a \cdot a \ldots . a$ (n veces) a se dice **base** y n se dice **exponente**

Propiedades

1) Si $a \neq 0$, entonces $a^0 = 1$ — Base no nula y exponente 0
2) $a^n.a^m = a^{n+m}$ — Producto de Potencias de la misma base
3) $a^n:a^m = a^{n-m}$, si n>m — Cociente de potencias de la misma base
4) $(a.b)^n = a^n.b^n$ — Potencia de un producto
5) $(a:b)^n = a^n:b^n$ — Potencia de un cociente
6) $(a^n)^m = a^{n.m}$ — Potencia de Potencia

Ejemplos

1) $(-3)^0 = 1$
2) $5^4.5^{-3} = 5^{4+(-3)} = 5$
3) $(-2)^7 : (-2)^5 = (-2)^2 = 4$
4) $(-3 . 10)^4 = (-3)^4 . 10^4 = 810000$
5) $(-8 : 2)^3 = (-8)^3 : 2^3 = -64$
6) $[(-1)^3]^5 = (-1)^{15} = -1$

Radicación

Sean $a \in Z$, $n \in N$,se define la **raíz enésima** $\sqrt[n]{a}$ como el número que elevado a la potencia n da como resultado $a \sqrt[n]{a} = b \Leftrightarrow b^n = a$ a se llama **radicando** y n **índice**

La radicación de números enteros no siempre es entero.

La radicación goza de las siguientes **propiedades**, siempre que las raíces involucradas estén definidas

1) $\sqrt[n]{a.b} = \sqrt[n]{a} \,.\, \sqrt[n]{b}$ Raíz de un producto

2) $\sqrt[n]{a:b} = \sqrt[n]{a} : \sqrt[n]{b}$ Raíz de un cociente

3) $\sqrt[m]{\sqrt[n]{a}} = \sqrt[n.m]{a}$ Raíz de raíz

Ejemplos

1) $\sqrt[3]{27.1000} = \sqrt[3]{27} \,.\, \sqrt[3]{1000} = 3.10 = 30$

2) $\sqrt{64:4} = \sqrt{64} : \sqrt{4} = 8:2 = 4$

3) $\sqrt[3]{\sqrt{64}} = \sqrt[6]{64} = 2$

Cuidado !!!

La potenciación y la radicación no son distributivas ni respecto de la suma ni respecto de la resta

Éste es un error muy frecuente entre los estudiantes del nivel medio. Por ello proponemos comparar los siguientes cálculos

sí !!!	no...!!!
Cálculos correctos	Cálculos incorrectos
$(2+3)^2 = 5^2 = 25$	$(2+3)^2 = 2^2 + 3^2 = 4 + 9 = 13$
$(7-4)^2 = 3^2 = 9$	$(7-4)^2 = 7^2 - 4^2 = 49 - 16 = 33$
$\sqrt{9+16} = \sqrt{25} = 5$	$\sqrt{9+16} = \sqrt{9} + \sqrt{16} = 3+4 = 7$
$\sqrt{25-16} = \sqrt{9} = 3$	$\sqrt{25-16} = \sqrt{25} - \sqrt{16} = 5-4 = 1$

CONTENIDO DISCIPLINAR: MATEMÁTICA

En el caso de tener expresiones algebraicas (expresiones que combinan números y letras) puede aplicarse, de ser necesario, la definición de potenciación y así encontrar una expresión algebraica equivalente

Productos notables

Las siguientes expresiones resultan de aplicar la definición de potenciación y las propiedades de la suma y el producto. Reciben el nombre de productos notables

$(a+b)^2 = a^2 + 2ab + b^2$ $\qquad$ $(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$

$(a-b)^2 = a^2 - 2ab + b^2$ $\qquad$ $(a-b)^3 = a^3 - 3a^2b + 3ab^2 - b^3$

$a^2 - b^2 = (a+b)(a-b)$

Ejercicios resueltos

1) Realizar los siguientes cálculos

a) $\sqrt[3]{-1}(-1)^3 + (-2)(-2)^3 - \sqrt{1+\sqrt{9}} + (-3)^5 : \sqrt[3]{-27}$

b) $(-48:12)^2 - [(-22):(-11)]^2 - [(-2)^2]^3 + (-3^0)^{10} - [2.(-5)]^2$

2) Aplicar propiedades para transformar las siguientes expresiones en otras equivalentes

a) $[2\,(-a).(-a)^3]^3 : (a+a)^2$ $\qquad$ b) $[2(a+b)]^3 : (a+b)$

c) $(a+2b).(a-2b)$ $\qquad$ d) $\sqrt{16(2a-b)^7} : \sqrt{2a-b}$

Respuestas:

a) $\sqrt[3]{-1}(-1)^3 + (-2)(-2)^3 - \sqrt{1+\sqrt{9}} - (-3)^2 : \sqrt[3]{-27}$

La expresión tiene cuatro términos, y los cálculos en cada término son

$\sqrt[3]{-1}(-1)^3 = (-1)(-1) = 1$ $\qquad$ $(-2)(-2)^3 = (-2)^4 = 16$

$\sqrt{1+\sqrt{9}} = \sqrt{1+3} = \sqrt{4} = 2$ $\qquad$ $(-3)^2 : \sqrt[3]{-27} = (-3)^2 : (-3) = (-3)^1 = -3$

Entonces

$\sqrt[3]{-1}(-1)^3 + (-2)(-2)^3 - \sqrt{1+\sqrt{9}} - (-3)^2 : \sqrt[3]{-27} = 1+16-2-(-3) = 18$

b) $(-48:12)^2 - [(-22):(-11)]^2 - [(-2)^2]^3 + (-3^0)^{11} - [2.(-5)]^2$

Los cálculos auxiliares son:

$(-48:12)^2 = (-4)^2 = 16$ $\qquad$ $[(-22):(-11)]^2 = 2^2 = 4$

$[(-2)^2]^3 = (-2)^6 = 64$ $\qquad$ $(-3^0)^{11} = (-1)^{11} = -1$ $\qquad$ $[2.(-5)]^2 = [-10]^2 = 100$

Entonces $(-48:12)^2 - [(-22):(-11)]^2 - [(-2)^2]^3 + (-3^0)^{11} - [2.(-5)]^2$ = 16-4-64-1-100=-153

2)

a) $[2\,(-a).(-a)^3]^3 : (a+a)^2 = [2\,(-a)^4]^3 : (2a)^2 = 2^3 \cdot a^{12} : (2^2 \cdot a^2) = 2\,a^{10}$

b) $[2(a+b)]^3 : (a+b) = 2^3 (a+b)^3 : (a+b) = 2^3 (a+b)^2 = 8(a^2+2ab+b^2)$

c) $(a+2b).(a-2b) = a^2 - 4b^2$

d) $\sqrt{16(2a-b)^7} : \sqrt{2a-b} = \sqrt{16(2a-b)^7 : (2a-b)} = \sqrt{16(2a-b)^6} = 4(2a-b)^3$

CONTENIDO DISCIPLINAR: MATEMÁTICA

NUMEROS RACIONALES

Dividir es repartir en partes iguales!!!
Un grupo de 6 amigos juega a las cartas con un mazo de 52cartas.
El juego consiste en repartir todas las cartas y dejar el resto en el centro de la mesa. ¿Cuántas cartas le corresponden a cada uno? ¿Cuántas cartas quedan en el centro?¡Tu puedes deducir la respuesta!¿Y si se quiere repartir pero el dividendo es menor que el divisor? Por ejemplo: Juana quiere repartir 1 barra de chocolate entre sus 3 amigos.
Entonces Juana da un tercio de chocolate a cada uno.

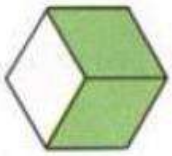
Definición

Los **Números Racionales** son los números que se pueden escribir como el cociente de dos enteros. Esto es, los que se pueden expresar como fracción. En símbolos

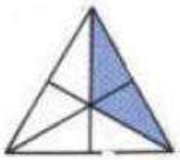
$$Q = \{ \frac{a}{b} \ / \ a,b \in Z \ \ y \ \ b \neq 0 \}$$

Los números racionales representan partes de un todo
Las partes sombreadas de los siguientes objetos están representadas por números Racionales

$\frac{5}{10}$



$\frac{2}{3}$



$\frac{2}{6}$

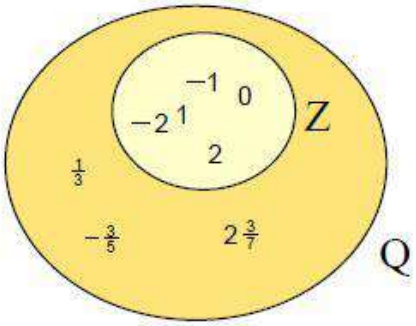
$\frac{6}{10}$

Observe que:

Si b = 1 o b = -1 , $\frac{a}{1} = a$ y $\frac{a}{-1} = -a$ son enteros.

Entonces "Todos los enteros son racionales"

Es decir $Z \subseteq Q$



Operaciones en Q

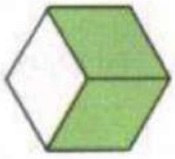
	Definición	Propiedad	Ejemplos
Suma y resta	1) $\frac{a}{b} \pm \frac{c}{b} = \frac{a \pm c}{b}$ 2) $\frac{a}{b} \pm \frac{c}{d} = \frac{a}{b}\frac{d}{d} \pm \frac{c}{d}\frac{b}{b} = \frac{ad \pm cb}{bd}$	La suma y resta de números racionales es siempre otro racional Si $u \in Q$ y $v \in Q$, entonces $u+v \in Q$ y $u-v \in Q$	1) $-\frac{3}{7} + \frac{9}{7} = \frac{6}{7}$ 2) $\frac{2}{3} - \frac{7}{4} = \frac{2}{3}\cdot\frac{4}{4} - \frac{7}{4}\cdot\frac{3}{3} = \frac{8}{12} - \frac{21}{12} = -\frac{13}{12}$

CONTENIDO DISCIPLINAR: MATEMÁTICA

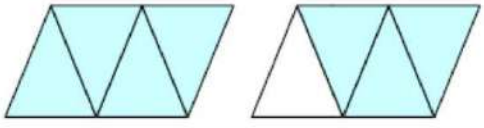
Producto y división	1) $\frac{a}{b}.\frac{c}{d}=\frac{a.c}{b.d}$ 2) $\frac{a}{b}:\frac{c}{d}=\frac{a.d}{b.c}$	El producto de números racionales es siempre otro racional. La división no se puede hacer en el caso de divisor nulo Si $u \in Q$ y $v \in Q$, entonces $u.v \in Q$ y $u:v \in Q$ si $v \neq 0$	1) $-\frac{3}{8}.\frac{4}{9}=-\frac{1}{6}$ 2) $\frac{15}{7}:\frac{-5}{2}=-\frac{6}{7}$
Potenciación y Radicación	Las definiciones son las mismas que las mencionadas para números enteros	Se añaden las siguientes Si $u \in Q$ y $m,n \in Z$ 1) $u^{-1}=\frac{1}{u}$, u^{-1} es el inverso de u 2) $u^{-n}=\frac{1}{u^n}$ 3) $u^{\frac{m}{n}}=\sqrt[n]{u^m}$	1) $\left(\frac{3}{4}\right)^{-1}=\frac{4}{3}$ 2) $2^{-3}=\frac{1}{2^3}=\frac{1}{8}$ 3) $8^{\frac{2}{3}}=\sqrt[3]{8^2}=$ $=\left(\sqrt[3]{8}\right)^2=4$

Notación decimal

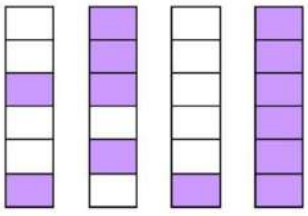
$\frac{5}{10}=\frac{1}{2}=\frac{2}{4}=\frac{3}{6}=0.5$ → decimal exacto



$\frac{2}{3}=\frac{4}{6}=\frac{6}{9}=0.666....=0.\hat{6}$ → decimal periódico puro, de periodo 6



$1+\frac{3}{4}=1\frac{3}{4}=1.75$ → decimal exacto



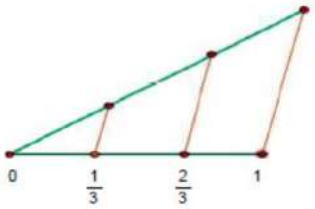
$\frac{2}{6}+\frac{4}{6}+\frac{1}{6}+1=\frac{7}{6}+1=\frac{13}{6}=2.1\hat{6}$ → decimal periódico mixto, de periodo 6 y anteperiodo 1

CONTENIDO DISCIPLINAR: MATEMÁTICA

Representación de los números racionales sobre la recta numérica

Aplicando el Teorema de Thales es posible ubicar a cada número racional de una manera exacta

Ejemplo: Para ubicar a los números $\frac{1}{3}$ y $\frac{2}{3}$ se toma una recta auxiliar con origen en 0 y sobre ella se marcan 3 segmentos cualesquiera pero iguales. Luego se une el extremo del último segmento con el número 1 y se trazan paralelas a esta línea. De esa manera la primera unidad sobre la recta numérica queda dividida en tres partes iguales y quedan determinado los números $\frac{1}{3}$ y $\frac{2}{3}$



"Todo número racional puede expresarse en notación decimal ya sea exacta o infinita periódica"

Cada número racional expresado en notación decimal está compuesto de dos partes

parte entera parte decimal

$-3,1\hat{2}$

período

anteperíodo

Conversión de la forma decimal a la forma fraccionaria

Forma decimal		Regla	Ejemplo
Exacta		En el numerador se coloca el número sin comas y en el denominador se coloca el 1 seguido de tantos ceros como cifras decimales tenga el número	$0,23 = \frac{23}{100}$ $1,005 = \frac{1005}{1000}$
Periódicas	Puras	En el numerador se coloca la diferencia entre la expresión sin la coma y la parte anterior al periodo y como denominador tantos 9 como cifras tiene el periodo	$0,232323\ldots = \frac{23}{99}$ $12,\hat{7} = \frac{127-12}{9} = \frac{115}{9}$
	Mixtas	En el numerador, la diferencia entre la expresión sin la coma y la parte anterior al periodo y en el denominador tantos 9 como cifras tiene el periodo y tantos 0 como cifras tenga el anteperiodo	$0,65\hat{8} = \frac{658-65}{900} = \frac{593}{900}$ $1,02525\ldots = \frac{1025-10}{990} = \frac{1015}{990}$

Q es un conjunto denso

Entre dos números racionales hay infinitos números racionales. Esta afirmación podría justificarse sencillamente si tenemos en cuenta que la suma de racionales es siempre otro racional, el promedio será otro racional y estará comprendido entre ellos. Podríamos continuar indefinidamente el procedimiento de promediar dos números racionales encontrando siempre que hay otro racional entre dos racionales por más próximos que estén. Por ello decimos que **Q es un conjunto denso**

NUMEROS IRRACIONALES

Todos los números racionales están representados por puntos sobre la recta numérica pero, ¿todos los puntos de la recta son representaciones de números racionales? La respuesta es NO!!! Existen otros números que junto a los racionales completan a la recta numérica. Ellos son los números irracionales

Definición

Los **Números Irracionales** son los números que no se pueden expresar como fracción. En símbolos

$$I = \{x \ / \ x \text{ no se puede expresar como fracción}\}$$

Convertidos a la notación decimal son números con infinitas cifras **no periódicas**

Ejemplos

Los siguientes son números irracionales famosos. Están redondeados en la 5ta cifra decimal, con lo cual se obtiene un valor aproximado bastante aceptable

a) El número PI: $\pi \cong 3.14159$ b) El número e $\cong 2.71828$

c) El número de oro: $\phi = \frac{1+\sqrt{5}}{2} \cong 1.61803$

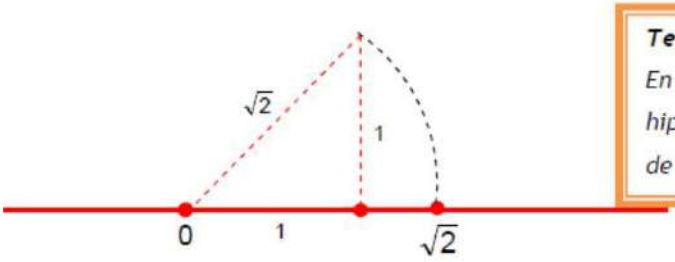
d) Raíces no exactas como ser : $\sqrt{2} \cong 1.41421$; $\sqrt{3} \cong 1.73205$; $\sqrt[3]{3} \cong 1.44225$

Ubicación exacta de $\sqrt{2}$

Con ayuda del Teorema de Pitágoras podemos ubicar de manera exacta a $\sqrt{2}$.

Si construimos un triángulo rectángulo de catetos unitarios, la hipotenusa mide $\sqrt{2}$

Luego con ayuda de un compás trasladamos la medida de la hipotenusa a la recta real



Teorema de Pitágoras

En todo triángulo rectángulo el cuadrado de la hipotenusa es igual a la suma de los cuadrados de los catetos.

Operando con números irracionales

CONTENIDO DISCIPLINAR: MATEMÁTICA

Las operaciones de suma, diferencia, producto, cociente y potenciación de números Irracionales no siempre arrojan como resultado a otro irracional. **Algunas veces los resultados son racionales!!**

Ejercicios resueltos

1) Realizar los siguientes cálculos y concluir de que naturaleza es el resultado encontrado:

a) $\pi + (-\pi)$ b) $\sqrt{2}+3\sqrt{2}$ c) $\sqrt{2}.\sqrt{8}$

d) $2^{\frac{2}{3}}$ e) $(\sqrt{2})^2$ f) $\sqrt[6]{27}$

2) Simplificar

a) $\sqrt{5}+\sqrt{45}-\sqrt{80}$ b) $\sqrt{175}-\sqrt{252}+3\sqrt{7}$

c) $\dfrac{\sqrt{24}.\sqrt{2}}{\sqrt{16^2+176}}$ d) $\dfrac{\sqrt{12}-\sqrt{48}}{\sqrt[3]{-8}}$

Respuestas:

1 a) $\pi + (-\pi) = \pi - \pi = 0 \in Q$

b) $\sqrt{2}+3\sqrt{2}=4\sqrt{2} \in I$

c) $\sqrt{2}.\sqrt{8}=\sqrt{16}=4 \in Q$

d) $2^{\frac{2}{3}}=(\sqrt{2})^3=(\sqrt{2})^2\sqrt{2}.=2\sqrt{2} \in I$

e) $(\sqrt{2})^2=2 \in Q$

f) $\sqrt[6]{27}=\sqrt[3]{\sqrt{27}}=\sqrt{\sqrt[3]{27}}=\sqrt{3} \in I$

2) a) $\sqrt{5}+\sqrt{45}-\sqrt{80}=\sqrt{5}+\sqrt{3^2.5}-\sqrt{2^4.5}=\sqrt{5}+3\sqrt{5}-4\sqrt{5}=4\sqrt{5}-4\sqrt{5}=0$

b) $\sqrt{175}-\sqrt{252}+3\sqrt{7}=\sqrt{5^2.7}-\sqrt{2^2.3^2.7}+3\sqrt{7}=5\sqrt{7}-6\sqrt{7}+3\sqrt{7}=2\sqrt{7}$

c) $\dfrac{\sqrt{24}.\sqrt{2}}{\sqrt{16^2+176}}=\dfrac{\sqrt{2^3.3}.\sqrt{2}}{\sqrt{432}}=\dfrac{\sqrt{2^3.3}.\sqrt{2}}{\sqrt{2^4.3^3}}=\sqrt{\dfrac{2^3.3.2}{2^4.3^3}}=\sqrt{\dfrac{1}{3^2}}=\dfrac{1}{3}$

d) $\dfrac{\sqrt{12}-\sqrt{48}}{\sqrt[3]{-8}}=\dfrac{\sqrt{2^2.3}-\sqrt{2^4.3}}{\sqrt[3]{-2^3}}=\dfrac{2\sqrt{3}-2^2\sqrt{3}}{-2}=\dfrac{-2\sqrt{3}}{-2}=\sqrt{3}$

¿Y si necesitáramos expresar a los números irracionales en forma decimal? Usamos las primeras cifras decimales. De ese modo se obtienen valores aproximados de los números irracionales. Entonces siempre se comete un error al tomar la notación decimal de un número irracional y el error cometido es menor que 1 unidad del orden de la última cifra conservada.

Ejemplo

Sabemos que $\pi = 3,14159265358979323846264338327950288419716939937510582 0...$

Las aproximaciones del número π con una, dos, tres, cuatro y cinco cifras decimales son

$\pi \cong 3,1$, con un error $\varepsilon < 0,1$

$\pi \cong 3,14$, con un error $\varepsilon < 0,01$

$\pi \cong 3,141$, con un error $\varepsilon < 0,001$

$\pi \cong 3,1415$, con un error $\varepsilon < 0,0001$

$\pi \cong 3,14159$, con un error $\varepsilon < 0,00001$

CONTENIDO DISCIPLINAR: MATEMÁTICA

Racionalización

Si las raíces aparecen en el denominador, en muchos casos es necesario eliminarla. A este proceso se lo conoce con el nombre de **Racionalización de denominadores**

Ejemplos

$$\frac{-3}{\sqrt{12}} \quad ; \quad \frac{\sqrt{8}}{\sqrt[3]{16}} \quad ; \quad \frac{3\sqrt{2}}{\sqrt{2}-1} \quad ; \quad \frac{\sqrt{2}+\sqrt{3}}{\sqrt{2}-\sqrt{3}}$$

Primer Caso: Un único término con raíz cuadrada en el denominador

Se multiplica y divide por la raíz presente en el denominador

Ejemplo:

$$\frac{-3}{\sqrt{12}} = \frac{-3}{\sqrt{12}}\frac{\sqrt{12}}{\sqrt{12}} = \frac{-3\sqrt{12}}{12} = -\frac{\sqrt{12}}{4} = -\frac{2\sqrt{3}}{4} = -\frac{\sqrt{3}}{2}$$

Segundo Caso: Un único término con raíz mayor que 2 en el denominador

Se multiplica y divide por la raíz presente en el denominador elevada a un exponente conveniente

Ejemplo:

$$\frac{\sqrt{8}}{\sqrt[3]{16}} = \frac{\sqrt{2^3}}{\sqrt[3]{2^4}} = \frac{2\sqrt{2}}{2\sqrt[3]{2}} = \frac{\sqrt{2}}{\sqrt[3]{2}}\frac{\sqrt[3]{2^2}}{\sqrt[3]{2^2}} = \frac{\sqrt{2}\sqrt[3]{2^2}}{\sqrt[3]{2^3}} = \frac{2^{\frac{1}{2}}.2^{\frac{2}{3}}}{2} = \frac{2^{\frac{7}{6}}}{2} = 2^{\frac{1}{6}} = \sqrt[6]{2}$$

Tercer Caso: En el denominador suma o resta de términos que contienen raíces cuadradas. Se multiplica y divide por el conjugado del denominador

Ejemplo:

a) $$\frac{3\sqrt{2}}{\sqrt{2}-1} = \frac{3\sqrt{2}}{\sqrt{2}-1}\frac{\sqrt{2}+1}{\sqrt{2}+1} = \frac{3\sqrt{2}\left(\sqrt{2}+1\right)}{\sqrt{2}^2-1^2} = \frac{6+3\sqrt{2}}{1} = 6+3\sqrt{2}$$

b) $$\frac{\sqrt{2}+\sqrt{3}}{\sqrt{2}-\sqrt{3}} = \frac{\sqrt{2}+\sqrt{3}}{\sqrt{2}-\sqrt{3}}.\frac{\sqrt{2}+\sqrt{3}}{\sqrt{2}+\sqrt{3}} = \frac{2+2\sqrt{2}\sqrt{3}+3}{2-3} = \frac{5+2\sqrt{6}}{-1} = -5-2\sqrt{6}$$

NUMEROS REALES

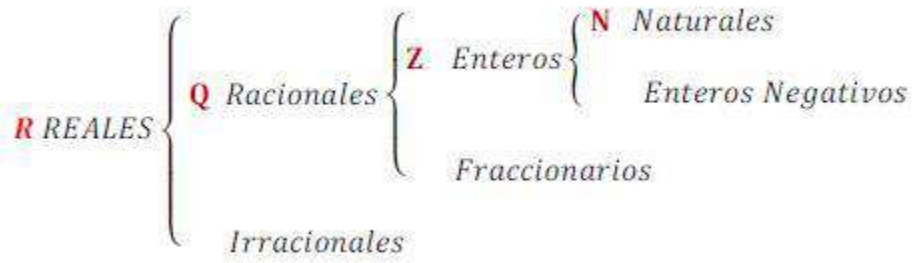
Entre los racionales y los irracionales se completa la recta numérica. Es decir ya no queda ningún punto sobre la recta al que no le corresponda ya sea un número racional o un número irracional. Es por ello que se considera que si se unen los dos conjuntos, esto es, Racionales más Irracionales se forma un nuevo conjunto **Definición**

El conjunto de los **Números Reales** es la unión del conjunto de los Racionales al conjunto de los Irracionales. Simbólicamente

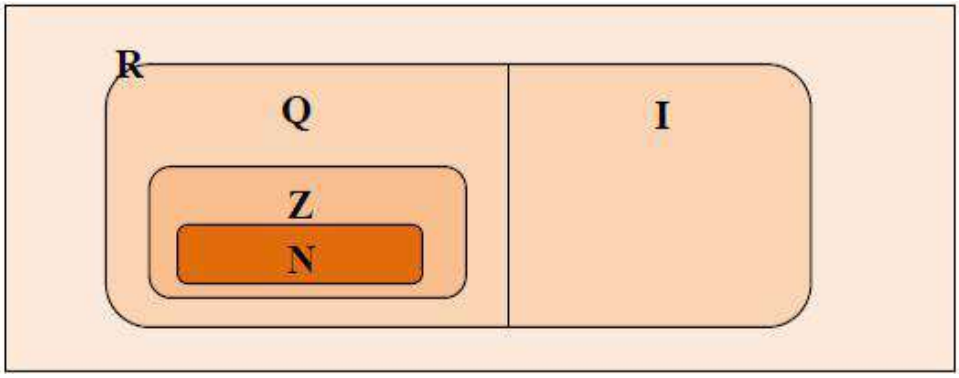
$$\mathbf{R} = \mathbf{Q} \cup \mathbf{I}$$

CONTENIDO DISCIPLINAR: MATEMÁTICA

A la recta numérica se le dice recta real pues en ella se representan a todos los números reales y, viceversa, todo punto de la recta es la representación de un real.
El conjunto **R** también tiene la propiedad de ser **denso.** De acuerdo a la definición se tiene el siguiente cuadro:



En un diagrama de Venn, se observa la relación entre los conjuntos



Notación científica

Cuando manejamos números muy grandes o muy pequeños tenemos dificultad para interpretarlos y para introducirlos en algunas calculadoras. Es usual, para ellos, representarlos mediante notación científica. Se dice que **un número está expresado en notación científica** cuando se escribe como el producto de un número mayor que 1 y menor que 10, multiplicado por una potencia entera de diez.

Ejemplos

Escribir los siguientes números en notación científica

9.800.000.000.000 ; 321.567.809.121.324

0,0000000000112 ; 0,00000000000134532

Respuestas

$9.800.000.000.000 = 9{,}8 \cdot 10^{12}$

$321.567.809.121.324 \cong 3{,}21.10^{14}$

$0{,}0000000000112 = 1{,}12 \cdot 10^{-11}$

$0{,}00000000000134532 \cong 1{,}34 \cdot 10^{-12}$

El conjunto R tiene estructura algebraica de Campo o Cuerpo

El conjunto R tiene estructura de **Campo** o **Cuerpo** pues las operaciones de suma y producto de números reales cumplen los siguientes axiomas:
Si *x, y, z* € R, entonces: La suma y el producto son operaciones **cerradas**

CONTENIDO DISCIPLINAR: MATEMÁTICA

$X + y \in R$ $(x.y) \in R$

La suma y el producto son operaciones **conmutativas**

$x + y = y + x$ $x.y = y.x$

La suma y el producto son operaciones **asociativas**

$(x+y) + z = x + (y+z)$ $(x.y).z = x.(y.z)$

El producto es **distributivo** respecto a la suma
$x.(x+z) = x.y + x.z$

Existen números reales que son **neutros** respecto de la suma y el producto
0 es el neutro respecto de la suma pues $x+0 = x$ 1 es el neutro respecto del producto pues $x.1 = x$
Todos los números reales tienen **opuesto** y, excepto el 0, todos tienen **recíproco**
$-x$ se dice inverso aditivo u opuesto de x
$1/x$ se dice inverso multiplicativo o recíproco de x

La división de números reales no goza de las propiedades conmutativas ni asociativa.

Ejemplos:

1) $2:3 \neq 3:2$

2) $6:(3:2) \neq (6:3):2$

Pero goza de la propiedad distributiva a izquierda.

Esto es: Sólo es válido distribuir el divisor en las sumas o restas presentes en el dividendo.

$$(a \pm b):c = a:c \pm b:c$$

$$\frac{a \pm b}{c} = \frac{a}{c} \pm \frac{b}{c}$$

Ejemplos:

1) $\frac{\sqrt{2}+2}{2} = \frac{\sqrt{2}}{2} + 1$

2) $\frac{6a-4b}{2} = \frac{6a}{2} - \frac{4b}{2} = 3a - 2b$

CONTENIDO DISCIPLINAR: MATEMÁTICA

Orden en el conjunto R

R es un conjunto ordenado. Esto es, dados dos números reales ha y b vale una y solo una de las siguientes afirmaciones
a <b, a > b o a = b

Propiedades de la Igualdad en R

1) Si sumamos o multiplicamos a ambos miembros de una igualdad una misma constante se obtiene otra igualdad

Si a = b, entonces a + c = b + c

Si a = b, entonces a.c = b.c

Ejemplos:

Como $4 = \sqrt{2}.\sqrt{8}$, entonces se tiene que $5 = \sqrt{2}.\sqrt{8}.+1$ y $1 = \frac{1}{4}\sqrt{2}.\sqrt{8}$

2) Si sumamos o multiplicamos miembro a miembro dos igualdades se obtiene otra igualdad

Si a = b y c = d, entonces a + c = b + d
Si a = b y c = d, entonces a. c = b. d

Ejemplos:

$4 = \sqrt{2}.\sqrt{8}$ y $8 = \sqrt{8}.\sqrt{8}$ $\Rightarrow$ $12 = \sqrt{8}(\sqrt{2} + \sqrt{8})$ y $32 = \sqrt{2}(\sqrt{8})^3$

Propiedades de la desigualdad

1) **Si a ambos miembros de una desigualdad se suma una misma constante , la desigualdad se mantiene**

Si a < b, entonces a+c < b+c

Ejemplo:

$1 < \sqrt{2}.$ $\Rightarrow$ $3 < 2+\sqrt{2}$

2) Si a ambos miembros de una desigualdad se multiplica por una misma constante positiva la desigualdad se mantiene

Si a < b y c > 0, entonces a.c < b.c

Ejemplo: $1 < \sqrt{2}.$ $\Rightarrow$ $2 < 2\sqrt{2}$

3) Si a ambos miembros de una desigualdad se multiplica por una misma constante negativa la desigualdad cambia de sentido

Si a < b y c < 0, entonces a.c>b.c

Ejemplo: $1 < \sqrt{2}.$ $\Rightarrow$ $-2 > -2\sqrt{2}$

Intervalos

CONTENIDO DISCIPLINAR: MATEMÁTICA

A menudo se trabaja con subconjuntos de números reales que representan semirrectas o segmentos de recta. La notación de Intervalos es muy conveniente

Intervalo abierto	$(a, b) = \{x \in R / a < x < b\}$
Intervalo cerrado	$[a, b] = \{x \in R / a \leq x \leq b\}$
Intervalos semiabiertos	$(a, b] = \{x \in R / a < x \leq b\}$
	$[a, b) = \{x \in R / a \leq x < b\}$
Intervalos infinitos	$[a, \infty) = \{x \in R / x \geq a\}$ $(a, \infty) = \{x \in R / x > a\}$
	$(-\infty, a] = \{x \in R / x \leq a\}$ $(-\infty, a) = \{x \in R / x < a\}$
	$(-\infty, \infty) = R$

Ejemplos

Dadas las siguientes desigualdades, expresarlas en notación de intervalos y grafique en la recta real

a) $-2 < x < 3$ b) $-2 \leq x \leq 3$ c) $-2 \leq x < 3$ d) $-2 < x \leq 3$

e) $x > -2$ f) $x \geq -2$ g) $x < -2$ h) $x \leq -2$

Respuestas

a) $(-2, 3)$

b) $[-2, 3]$

c) $[-2, 3)$

d) $(-2, 3]$

e) $(-2, \infty)$

f) $[-2, \infty)$

g) $(-\infty, -2)$

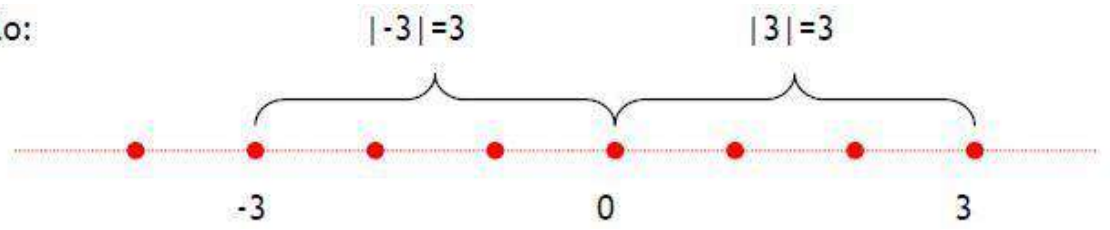
h) $(-\infty, -2]$

Modulo o Valor absoluto de un número real

El valor absoluto o módulo de un número mide la distancia desde el número al origen. Se denota con |a|.

CONTENIDO DISCIPLINAR: Matemática

Ejemplo:



Propiedades

El valor absoluto de un número es siempre mayor o igual a cero $|a| \geq 0$
Los números opuestos tienen el mismo valor absoluto $|a| = |-a|$
El valor absoluto es distributivo respecto del producto $|a.b| = |a|.|b|$
El valor absoluto es distributivo respecto del cociente $|a:b| = |a|:|b|$

Distancia entre dos reales

Sean a y b números reales, entonces la distancia entre a y b se calcula como $|a - b|$

Ejemplos

La distancia entre -3 y 5 es $|-3-5| = |-8| = 8$

La distancia entre $\frac{13}{5}$ y $\frac{1}{10}$ es $\left|\frac{13}{5}-\frac{1}{10}\right| = \left|\frac{5}{2}\right| = 2{,}5$

La distancia entre $\sqrt{2}$ y 3 es $\left|\sqrt{2}-3\right| \approx \left|-1{,}585\right| = 1{,}585$

Relación entre el valor absoluto y la raíz cuadrada de un número real

Si $a \in R$, entonces $\sqrt{a^2} = |a|$

Ejemplos: $\sqrt{3^2} = |3| = 3$ y $\sqrt{(-3)^2} = |-3| = 3$

La séptima operación: Logaritmo de un número real

$$\log_b a = n \iff b^n = a \;;\; con \;\; a, b > 0 \;,\; b \neq 1$$

a es llamado número logaritmado, b es llamado base del logaritmo y n valor del logaritmo.

Ejemplos:

a) $\log_2 8 = 3$ porque $2^3 = 8$

b) $\log_{\frac{1}{3}} 3 = -1$ porque $\left(\frac{1}{3}\right)^{-1} = 3$

c) $\log_3 3 = 1$ y en general $\log_b b = 1$

d) $\log_5 1 = 0$ y en general $\log_b 1 = 0$

e) $\log_b(-4)$ no existe . ¿Porqué?

f) $\log_b 0$ no existe. ¿Porqué?

CONTENIDO DISCIPLINAR: Matemática

Propiedades del Logaritmo:

Siempre que los logaritmos involucrados estén definidos valen las siguientes propiedades

Logaritmo de un producto: $\log_b(m.n) = \log_b m + \log_b n$

Logaritmo de un cociente : $\log_b(m:n) = \log_b m - \log_b n$

Logaritmo de una potencia: $\log_b m^n = n.\log_b m$

Ejemplos:

Calcular usando definición y propiedades de logaritmo

a) $\log 20 + \log 5$ b) $\log_3 7 - \log_3 21$

c) $\log_2 4^{250}$ d) $\frac{1}{2}.\log_5 100 + \log_5 \frac{1}{2}$

Respuestas

a) $\log 20 + \log 5 = \log (20 . 5) = \log 100 = 2$

b) $\log_3 7 - \log_3 21 = \log_3 (7 : 21) = \log_3 1/3 = - 1$

c) $\log_2 4^{250} = 250 . \log_2 4 = 250 . 2 = 500$

d) $\frac{1}{2}.\log_5 100 + \log_5 \frac{1}{2} = \log_5 100^{1/2} + \log_5 \frac{1}{2} = \log_5 10 + \log_5 \frac{1}{2} =$

$\log_5 (10.\frac{1}{2}) = \log_5 5 = 1$

NUMEROS COMPLEJOS

Los números complejos son combinaciones algebraicas de números reales con números imaginarios. ¿Por qué surgen los números imaginarios?
Las raíces de índice par de radicando negativo no tienen respuesta en R. Para dar solución a este problema se crea el número j.

Definición:

CONTENIDO DISCIPLINAR: Matemática

Se define el **número j, unidad imaginaria** , como aquel cuyo cuadrado es -1 .

Esto es $j^2 = -1$

De este modo las raíces cuadradas de radicando negativo tienen solución

Ejemplos

a) Calcular: $\sqrt{-1}$; $\sqrt{-4}$; $\sqrt{-3}$; $\sqrt{-\frac{4}{3}}$

b) Calcular $(-j)^2$, $(2j)^2$, $(-2j)^2$, $(\sqrt{3}j)^2$, j^3 , j^4

Respuestas:

a) $\sqrt{-1} = j$; $\sqrt{-4} = 2j$; $\sqrt{-3} = \sqrt{3}j$; $\sqrt{-\frac{4}{3}} = \frac{2}{\sqrt{3}}j$

b) $(-j)^2 = (-j)(-j) = j^2 = -1$ $(2j)^2 = -4$

$(-2j)^2 = -4$ $(\sqrt{3}j)^2 = (\sqrt{3})^2 . j^2 = -3$

$j^3 = j^2 . j = (-1) . j = -j$ $j^4 = (j^2)^2 = (-1)^2 = 1$

Potencia enésima de la unidad imaginaria

Si n Є N, al dividir n en 4 puede expresarse como n = 4. q + r, donde q es el cociente y r es el resto. Entonces 0 ≤ r < 4 y la potencia enésima de j se calculan como:

$$j^n = j^{4q+r} = (j^4)^q . j^r = 1 . j^r = j^r$$

Ejemplos

$j^{103} = j^{4.25+3} = (j^4)^{25} . j^3 = 1^{25} . j^3 = j^3 = -j$

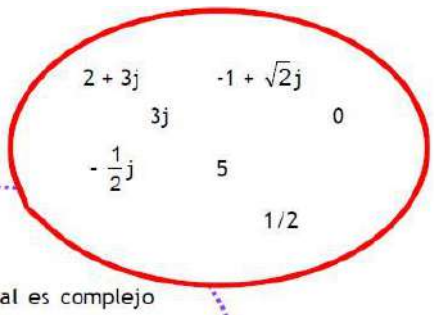
$j^{1012} = j^{4.253} = 1$

Definición

Se define al conjunto de los **Números Complejos** como **C = { z / z = a + bj , a Є R y b Є R }**

a se dice **componente real** y **b** se dice **componente imaginaria**

El conjunto C también tiene estructura de **Campo**, respecto de la suma y el producto



Observe que

Todo número real es complejo de parte imaginaria nula:

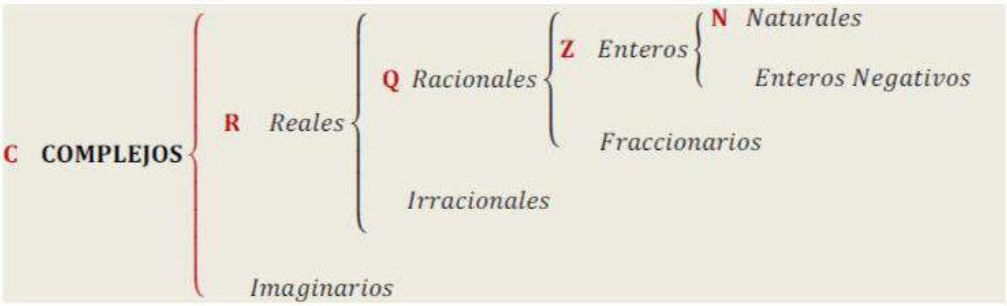
5= 5+ 0j

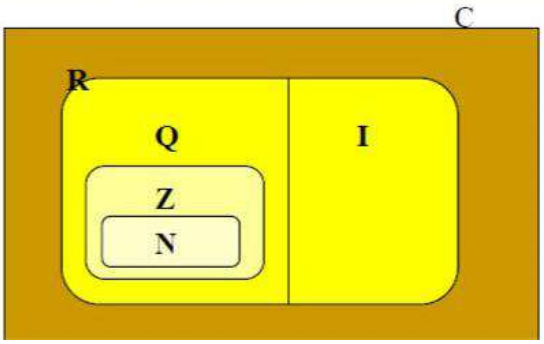
Todo número imaginario es complejo de parte real nula:

-2j = 0 - 2j

CONTENIDO DISCIPLINAR: Matemática

Las relaciones entre los conjuntos numéricos estudiados se muestran en las siguientes Figuras:





Todo número complejo está asociado a otros llamados opuesto y conjugado

Sea $Z = a + bi$, al número $-Z = -a - bj$ se le llama **opuesto** de Z

Sea $Z = a + bi$, al número $\bar{Z} = a - bj$ se le llama **conjugado** de Z

Ejemplos:

Si $z = 1 - 3j$, entonces $-z = -1 + 3j$ y $\bar{z} = 1 + 3j$

Si $z = -2j$, entonces $-z = 2j$ y $\bar{z} = 2j$

Si $z = -1$, entonces $-z = 1$ y $\bar{z} = -1$

Igualdad en C

Dos números complejos son iguales si y solo si sus componentes respectivas son iguales.
Esto es: $a + bj = c + dj$; $a = c \wedge b = d$

Ejercicios resueltos

Encontrar el valor de $k \in R$ para que se cumple que $z_1 = z_2$ siendo

a) $z_1 = 2 - 3kj$ y $z_2 = -k + 6j$

b) $z_1 = 1 + k + j$ y $z_2 = -3/2 + j$

Respuesta

a) $z_1 = z_2 \Leftrightarrow 2 = -k \text{ y } -3k = 6 \Rightarrow k = -2$

b) $z_1 = z_2 \Leftrightarrow 1 + k = -3/2 \text{ y } 1 = 1 \Rightarrow k = -5/2$

CONTENIDO DISCIPLINAR: Matemática

Operaciones en c:

Operación	Definición	Ejemplos
Suma y resta	(a+bj)+(c+dj) = (a+c)+(b+d)j (a+bj)-(c+dj) = (a-c)+(b-d)j	1) (2+3j)+($\sqrt{2}$-4j)= (2+$\sqrt{2}$)-j 2) (1+5j)-(3/4 - j) =1/4 +6j
Producto	(a+bj)+(c+dj) = a.c+adj+cbj+bdj^2= (ac-bd)+(ad+cb)j	(-1/2+j).(1-j) = -1/2 +1/2j +j -j^2 = (-1/2+1)+(1/2+1)j= 1/2 +3/2j
División	$\frac{a+bj}{c+dj}=\frac{a+bj}{c+dj}\cdot\frac{\overline{c+dj}}{\overline{c+dj}}=$ $\frac{a+bj}{c+dj}\cdot\frac{c-dj}{c-dj}=\frac{(ac-bd)+(bc+ad)j}{c^2+d^2}$	$\frac{1-2j}{-3+j}=\frac{(1-2j)(-3-j)}{(-3+j)(-3-j)}=$ $=\frac{(-3-2)+(-1+6)j}{(-3)^2+1^2}=\frac{1}{10}(-5+7j)$

Propiedades del conjugado:

La suma de un complejo y su conjugado es siempre real

$$Z+\overline{Z}=(a+bj)+(a-bj)=a+bj+a-bj=2a\in R$$

El producto de un complejo y su conjugado es siempre real

$$Z.\overline{Z}=(a+bj).(a-bj)=a^2+abj-abj-(bj)^2=a^2+b^2\in R$$

Ejercicios resueltos

Encontrar el valor de Z tal que

a) $Z=\frac{(1+j)(1-j)+2j}{(-2+j)^2}-j^3$ b) $Z=(1-j)^3:(\sqrt{2}+j)+j(-j)^{27}$

Respuestas

a) $Z=\frac{(1+j)(1-j)+2j}{(-2+j)^2}-j^3=\frac{(1+1)+2j}{4+4j+j^2}-(-j)=$

$=\frac{2+2j}{3+4j}+j=\frac{2+2j+j(3+4j)}{3+4j}=$

$=\frac{-2+5j}{3+4j}=\frac{(-2+5j)}{(3+4j)}\frac{(3-4j)}{(3-4j)}=$

$=\frac{-6+8j+15j+20}{9+16}=\frac{1}{25}(14+23j)$

b) $Z = (1-j)^3 : (\sqrt{2}j) + j(-j)^{27} = \frac{(1-j)^3}{\sqrt{2}j} - j.j^{27}$

$= \frac{-2-2j}{\sqrt{2}j} - j^{28} = \frac{-2-2j}{\sqrt{2}j} - 1$

$= \frac{(-2-2j)}{\sqrt{2}j}\frac{\sqrt{2}j}{\sqrt{2}j} - 1 = \frac{(-2-2j)\sqrt{2}j}{-2} - 1$

$= \frac{-2(1+j)\sqrt{2}j}{-2} - 1 = (1+j)\sqrt{2}j - 1$

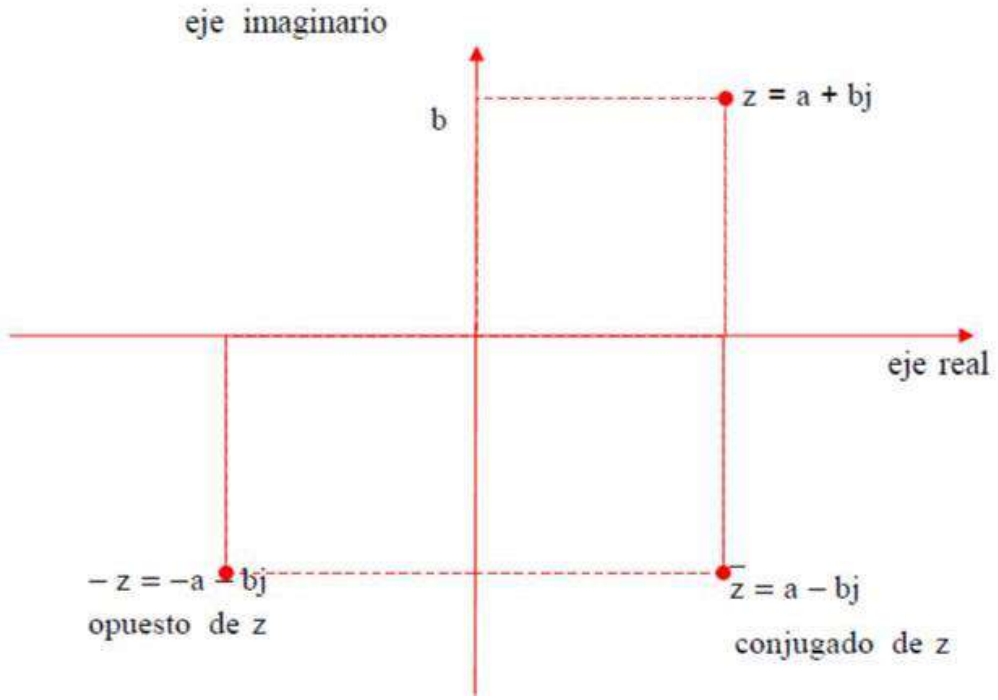
$= \sqrt{2}j - \sqrt{2} - 1$

Cálculo auxiliar:

$(1-j)^3 = 1 - 3j + 3j^2 - j^3 = 1 - 3$
$= -2 - 2j$

Representación gráfica de los números complejos

Todo número complejo z = a+bj se representa en el plano mediante el punto (a,b). Sobre el eje horizontal se representa a la componente real del complejo, por lo que a este eje se lo llama eje real. Sobre el eje vertical se representa a la componente imaginaria y por ello se lo llama eje imaginario 0.



C tiene estructura algebraica de Campo o Cuerpo

El conjunto C tiene estructura algebraica de Campo respecto de las operaciones de Suma y Producto pues en él se cumplen las propiedades de:

CONTENIDO DISCIPLINAR: Matemática

$\forall z_1, z_2, z_3 \in C$
La suma y el producto son **operaciones cerradas**

$$z_1 + z_2 \in C \qquad z_1.z_2 \in C$$

La suma y el producto son operaciones **conmutativas**

$$z_1 + z_2 = z_2 + z_1 \qquad z_1.z_2 = z_2.z_1$$

La suma y el producto son operaciones **asociativas**

$$(z_1 + z_2) + z_3 = z_1 + (z_2 + z_3) \qquad (z_1.z_2)z_3 = z_1.(z_2.z_3)$$

El producto es **distributivo** respecto a la suma

$$z_1.(z_2 + z_3) = z_1.z_2 + z_1.z_3$$

Existen números complejos que son **neutros** respecto de la suma y el producto 0 es el neutro respecto de la suma pues z + 0 = z 1 es el neutro respecto del producto pues z.1= z
Todos los números complejos tienen **opuesto** y, excepto el 0, todos tienen **recíproco**
–z se dice inverso aditivo u opuesto de z
1/z se dice inverso multiplicativo o recíproco de z

NÚMEROS PRIMOS.

Sea n Є *N*, con n>1, n es primo si y solo si tiene exactamente dos divisores positivos: 1 y n
Los primeros números primos son: 2 , 3 , 5 , 7 , 11, 13 , 17 , 19 , 23 , 29 , 31 , etc. Todo número natural puede descomponerse como producto de factores primos Ejemplos:
Expresar a 750, 480 y 1734 en su forma factoreada

750	2
375	3
125	5
25	5
5	5
1	

$750 = 2.3.5^3$

480	2
240	2
120	2
60	2
30	2
15	3
5	5
1	

$480 = 2^5 . 3 . 5$

1734	2
867	3
289	17
17	17
1	

$1734 = 2 . 3 . 17^2$

Máximo Común Divisor

Dados dos números enteros a y b.
Al número que es divisor de ambos y es el mayor de todos los divisores comunes se le llama máximo común divisor (mcd). El mcd(a,b) es igual al producto de todos los factores primos comunes entre a y b con su menor exponente

CONTENIDO DISCIPLINAR: Matemática

Mínimo Común Múltiplo

Al número que es múltiplo de ambos y es el menor de todos los múltiplos comunes se le llama mínimo común múltiplo (mcm). El mcm(a,b) es igual al producto de todos los factores primos comunes y no comunes con su mayor exponente Ejemplos

$mcd(250,480) = 2 \cdot 3 \cdot 5 = 30$ y $mcm(250, 480) = 2^5 \cdot 3 \cdot 5^3 = 12000$

$mcd(480, 1734) = 2 \cdot 3 = 6$ y $mcm(480, 1734) = 2^5 \cdot 3 \cdot 5 \cdot 17^2 = 138720$

CONTENIDO DISCIPLINAR: TECNOLOGÍA DE LA INFORMACIÓN y SISTEMAS OPERATIVOS

Introducción a la Informática: CONCEPTOS BÁSICOS

En la vida moderna las computadoras constituyen un componente esencial y, aunque no lo notemos, están en todas partes y son determinantes en nuestro modo de vida. Aún más, muchas veces nos damos cuenta de esto cuando dejan de funcionar. Pensemos por un momento en qué cosas está presente alguna forma de computadora: reloj despertador digital, radio, TV, reproductor de CD, agenda electrónica, cafetera automática, horno a microondas, encendido electrónico del auto, portón eléctrico de la cochera, teléfono celular, cajero automático, lector de tarjeta de ingreso al trabajo, ascensores automáticos, controles de seguridad del edificio, lavarropas automático, cámaras fotográficas, máquinas de juegos, expendedoras de comestibles, control de los semáforos, Smart TV, centrales telefónicas, aviones, aeropuertos, ... *casi todo* !!!!!

Es difícil imaginarse un día en el cual no utilicemos *alguno* de estos elementos.
¿Qué pasaría si todos ellos dejaran de funcionar simultáneamente?. Nuestra vida está relacionada con las computadoras, tanto por su operación como por su falta de funcionamiento. Y lo más sorprendente es que se hayan infiltrado tanto en la vida diaria en un tiempo tan corto...

¿QUÉ ES INFORMÁTICA?

La informática nace de la idea de ayudar al hombre en aquellos trabajos rutinarios y repetitivos, generalmente de cálculo y gestión, donde es frecuente la repetición de tareas. La idea es que una máquina puede realizarlos mejor, aunque siempre bajo la supervisión del hombre.

El término *Informática* se creó en Francia en 1962 bajo la denominación *Informatique*, y procede de la contracción de las palabras *Infor*mation auto*matique*.

Posteriormente fue reconocido por el resto de los países, siendo adoptado por España en 1968 bajo el nombre de Informática, que como puede deducirse fácilmente, viene de la contracción de las palabras *Infor*mación auto*mática*. En los países anglosajones se conoce con el nombre de *Computer Science*.
La informática se puede definir de diversas formas si bien todas ellas giran en torno a la misma idea. Dos de las más difundidas son:

Informática es la ciencia que estudia el tratamiento automático y racional de la información.

Informática *es la ciencia que estudia el análisis y resolución de problemas utilizando computadoras.*

La palabra ***ciencia*** se relaciona con una metodología fundamentada y racional para el estudio y resolución de los problemas.

La ***resolución de problemas*** utilizando las herramientas informáticas puede tener aplicaciones en áreas muy diferentes tales como biología, comercio, control industrial, administración, robótica, educación, arquitectura, diseño, etc.

Los temas propios de la ciencia Informática abarcan aspectos tales como la arquitectura física y lógica de las computadoras, las metodologías de análisis y diseño de sistemas de software, los lenguajes de programación, los sistemas operativos, la inteligencia artificial, los sistemas de tiempo real, el diseño y aplicación de bases de datos, etc.

CONTENIDO DISCIPLINAR: TECNOLOGÍA DE LA INFORMACIÓN y SISTEMAS OPERATIVOS

Es un concepto sinónimo de computación, y lo definiremos como conjunto de conocimientos científicos y de técnicas que hacen posible el tratamiento automático de la información por medio de computadoras. La informática combina los aspectos teóricos y prácticos de la ingeniería electrónica, teoría de la información, matemática, lógica y comportamiento humano. Los aspectos de la informática cubren desde la programación y la arquitectura informática hasta la inteligencia artificial y la robótica.

¿QUÉ ES UNA COMPUTADORA?

Una ***Computadora*** es una máquina digital y sincrónica, con cierta capacidad de cálculo numérico y lógico, controlada por un programa almacenado y con posibilidad de comunicación con el mundo exterior.

¿Qué significa esto?

- **Es d*igital*** porque dentro de la computadora las señales eléctricas que se manejan y la información que se procesa se representa en forma discreta, por medio de dos valores (0 y 1).

- Además se afirma que **es sincrónica**, es decir que realiza las operaciones coordinada por un reloj central que envía señales de sincronismo a todos los elementos que componen la computadora. Esto significa que todas las operaciones internas se realizan en instantes de tiempo predefinidos y coordinados con el reloj.

- Internamente posee una **capacidad de cálculo numérico y lógico,** en un subsistema denominado Unidad Aritmético-Lógica (UAL) ó en su acrónimo en idioma inglés ALU (Arithmetic & Logic Unit). Normalmente las operaciones que pueden realizarse en ella son muy simples (por ejemplo suma, disyunción, conjunción o comparación).

- El hecho que sea **controlada por programa** es quizás el punto más importante que diferencia a una computadora de una calculadora. Significa que internamente se tienen órdenes o instrucciones almacenadas, que la computadora podrá obtener, interpretar y ejecutar.

- Además, está **comunicada con el mundo exterior. Esto** significa que podrá realizar operaciones de ingreso o egreso de valores desde y hacia el mundo real, utilizando dispositivos periféricos (por ejemplo el teclado o el mouse para entrada de información y pantalla como salida). Debe mencionarse que el mundo real es *analógico* y no digital.

La computadora es una máquina que cambia información de una forma a otra: recibe información (entrada), la transforma y proporciona información (salida). Esta información puede presentarse de muchas formas, lo que convierte a la computadora en una máquina sumamente versátil, que es capaz desde liquidar impuestos hasta guiar el recorrido de una nave espacial. En cada caso las entradas y salidas son totalmente distintas, y en esto radica lo sorprendente de poder usar una computadora para ambas actividades.

Esta versatilidad está dada en que la máquina sea controlada por un *programa*, que establece las instrucciones que le indican a las partes físicas qué deben hacer para transformar los datos de entrada en la salida requerida. El programa controla todo el proceso, del principio al fin: podemos modificar su funcionamiento con solo cambiar el programa.

Una computadora es un dispositivo electrónico capaz de recibir un conjunto de instrucciones y ejecutarlas realizando cálculos sobre los datos numéricos, o bien compilando y correlacionando otros tipos de información.

CONTENIDO DISCIPLINAR: TECNOLOGÍA DE LA INFORMACIÓN y SISTEMAS OPERATIVOS

DATOS: conjunto de símbolos que representan la información de manera que se permita su procesamiento.

EQUIPOS (HARDWARE): es el conjunto de piezas físicas que integran una computadora: unidad central de proceso, placa base, periféricos y redes.

PERIFÉRICOS: son dispositivos encargados del ingreso, salida y almacenamiento de datos.

PROGRAMAS (SOFTWARE): contiene las instrucciones que le permiten al equipo físico realizar una tarea específica. Están entregados por varios archivos que realizan diversas funciones. Hay tres tipos de software: los sistemas operativos, los lenguajes de programación y las aplicaciones informáticas.

SISTEMAS OPERATIVOS: son el software básico que controla los recursos de hardware de la computadora. Sirven de enlace entre la computadora, las aplicaciones informáticas y los lenguajes. Realizan tres funciones principales:

- Coordinan y manejan el hardware de la computadora.
- Organizan los archivos en varios dispositivos de almacenamiento.
- Solucionan los errores de hardware y la pérdida de datos.

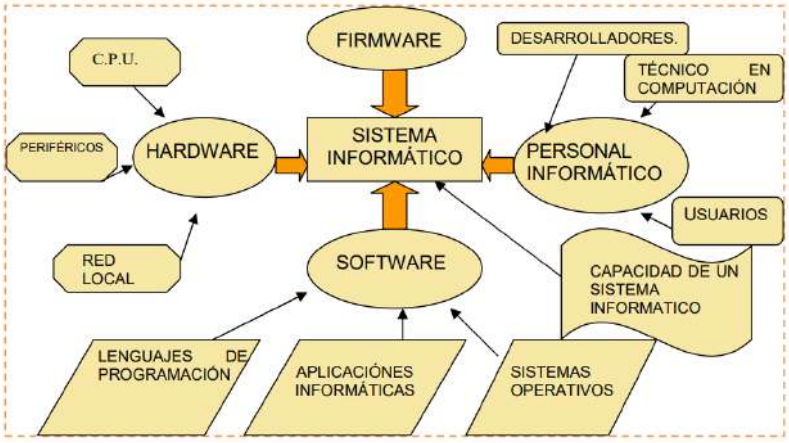
LENGUAJES DE PROGRAMACIÓN: son lenguajes artificiales, se utilizan para definir una serie de instrucciones, que representan las tareas que procesará una computadora. Al conjunto de instrucciones agrupadas en un archivo ejecutable se le conoce como *PROGRAMA*.

APLICACIONES INFORMÁTICAS: con el sistema operativo ya vienen incorporados programas (aplicaciones) para realizar las tareas sencillas, como la calculadora, reproductor de audio y video, navegador de internet, etc. Para realizar tareas más complejas o específicas, es necesario instalar otros programas (aplicaciones) desarrollados para dichas tareas. También llamado *SOFTWARE DE APLICACIÓN* realiza las funciones más comunes dentro de la casa, escuela u oficina. Son las aplicaciones básicas que todo usuario debe de conocer: (procesador de textos, hoja de cálculo, programa de presentaciones, administrador de bases de datos, editor gráfico, navegador de Internet, correo electrónico, agenda electrónica, etc.

FIRMWARE: es el software de sistema que reside en la memoria permanente de la computadora.

CONTENIDO DISCIPLINAR: TECNOLOGÍA DE LA INFORMACIÓN y SISTEMAS OPERATIVOS

PERSONAL INFORMÁTICO: son los usuarios del sistema informático de los desarrolladores, quienes diseñan el sistema y el personal que se encarga de mantenerlo en funcionamiento.



TIPOS DE SISTEMAS OPERATIVOS: una vez que se conoce que es un *SISTEMA OPERATIVO*, veamos que hay diferentes maneras de categorizarlos aparte del tipo de interfaz del usuario que utilizan.

Los sistemas operativos son diseñados con muchos objetivos en mente.

- Monotareas: se debe de esperar a terminar una tarea para iniciar otra.
- Multitareas: ejecuta varios programas al mismo tiempo.
- Multiusuarios: permite que más de un usuario utilice el sistema informático al mismo tiempo.
- Multiprocesos: sólo en sistemas informáticos que cuenten con dos o más procesos conectados entre sí. (linux y unix.)

INSTALADOR: es un archivo ejecutable necesario para instalar un programa en la PC. Este archivo puede estar acompañado de una serie de carpetas y otros archivos o ser un único archivo, dependiendo de la complejidad del programa. El *tipo* de un archivo *Instalador* es *Aplicación*.

ARCHIVOS Y CARPETAS

ARCHIVO: es un elemento que contiene información, como texto, imágenes o música. En el equipo, los archivos se representan mediante iconos, así resulta más sencillo reconocer un tipo de archivo con solo ver el icono. Cada tipo de archivo necesita de un programa compatible para ser abierto y editado. Un archivo de música, no se puede abrir con un programa editor de texto. Se necesita un programa de reproducción multimedia. Los archivos son las unidades básicas de almacenamiento que permiten a la computadora distinguir entre los diversos conjuntos de información. Aunque no siempre es el caso, un archivo se suele encontrar en un formato legible por los usuarios. Aun así, en un archivo se agrupan instrucciones, números, palabras o imágenes en unidades coherentes que el usuario puede recuperar, modificar, eliminar, guarda o enviar a un dispositivo de salida.

CARPETAS: una carpeta es un contenedor que se utiliza para almacenar archivos. Si tuviera miles de archivos en papel en el escritorio, sería prácticamente imposible encontrar uno concreto cuando lo necesitara. Por este motivo, la gente a menudo almacena los archivos en papel en archivadores. Las carpetas del equipo funcionan exactamente del mismo modo.

DIRECTORIO

CONTENIDO DISCIPLINAR: TECNOLOGÍA DE LA INFORMACIÓN y SISTEMAS OPERATIVOS

Un directorio en informática, es una organización jerárquica de nombres de archivos almacenados en un disco. El directorio superior se denomina directorio raíz, los directorios existentes dentro de otro directorio se denominan subdirectorios. Según la forma en que el sistema operativo soporte los directorios, los nombres de los archivos allí contenidos pueden verse y ordenarse de distinto modo como por ejemplo alfabéticamente, por fecha o por tamaño, o en forma de iconos en una interfaz gráfica de usuario. Lo que el usuario ve como directorio esta soporta en el sistema operativo en forma de tablas de datos, guardadas en el disco que contiene las características asociadas con cada archivo, así como la ubicación de éste dentro del disco.

PARTICIONES Y DISCOS LOCALES

DISCO RÍGIDO: es el dispositivo físico de almacenamiento que está instalado dentro del gabinete de la PC.

PARTICIONES: son divisiones lógicas que se realizan para asignar el espacio del disco rígido a unidades de almacenamiento utilizables por el usuario. Un disco puede tener una única partición o estar dividido en varias. Comúnmente se utilizan dos particiones, en la *primera* se almacena el SO y los programas y en la *segunda* los archivos del usuario.

UNIDAD DE DISCO DURO: es la forma en que el SO muestra al usuario las particiones asignadas. El nombre por defecto (modificable por el usuario) es Disco Local. Cada unidad lleva asignada una letra. Por defecto C, D, E.

COMPONENTES: Determinan el uso y rendimiento de las PCs. En el caso de que se desea adquirir o actualizar una PC de escritorio se debe tener en cuenta los siguientes componentes:

PROCESADOR: es el componente fundamental, realiza todos los cálculos y ejecuta todos los procesos, se debe tener en cuenta la cantidad de núcleos y velocidad (frecuencia de trabajo (GHz)).

MEMORIA RAM: todas las instrucciones, datos y programas abiertos, están almacenados allí mientras la PC está encendida. Según el uso de la Pc será necesaria cierta cantidad de memoria RAM (2GB, 4GB, 8GB).

TARJETA DE VIDEO: en el caso de usar la PC para trabajos de gráfica, edición de video, juegos de pc, se debe tener en cuenta la velocidad y la cantidad de memoria. Disco Rígido: es un dispositivo de almacenamiento permanente donde se guarda toda la información de nuestra computadora. El factor a tener en cuenta es la capacidad de almacenamiento que para un uso normal va de 500 GB a 1 TB (1000 Gb).

Aplicaciones de la informática

"El grado de inteligencia que atribuimos al comportamiento de algo está determinado tanto por nuestra propia capacidad y comprensión como por las propiedades del objeto que analizamos".
Alan Turing.

El universo de las aplicaciones informáticas es esencialmente multidisciplinario.

CONTENIDO DISCIPLINAR: TECNOLOGÍA DE LA INFORMACIÓN y SISTEMAS OPERATIVOS

Las aplicaciones que pueden desarrollarse con una computadora van desde un sistema de gestión comercial, administrativo, hasta sistemas expertos que ayudan en la toma de decisiones, diseño asistido, controladores de vuelo automáticos, máquinas jugadoras de ajedrez, etc.
En esta tarea están involucradas personas de distintas disciplinas: matemáticos, ingenieros e informáticos. Los matemáticos brindan las herramientas básicas para que tanto ingenieros como informáticos puedan desarrollar su labor. Por otro lado se encuentran los usuarios de las aplicaciones, que van desde especialistas que utilizan una determinada herramienta (economistas, docentes, músicos, médicos, arquitectos, etc.) hasta entusiastas que navegan por Internet o juegan con un simulador de vuelo.

Definición de software

Probablemente la definición más formal de software sea la siguiente:

> Es el conjunto de los programas informáticos, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.
>
> Extraído del estándar 729 del IEEE6[3]

Considerando esta definición, el concepto de software va más allá de los programas de cómputo en sus distintos estados: código fuente, binario o ejecutable; también su documentación, datos a procesar e información de usuario forman parte del software: es decir, abarca todo lo intangible, todo lo "no físico" relacionado.
El término «software» fue usado por primera vez en este sentido por John W. Tukey en 1957. En las ciencias de la computación y la ingeniería de software, el software es toda la información procesada por los sistemas informáticos: programas y datos.

Clasificación del software

Si bien esta distinción es, en cierto modo, arbitraria, y a veces confusa, a los fines prácticos se puede clasificar al software en tres grandes tipos:

Software de sistema

- Su objetivo es desvincular adecuadamente al usuario y al programador de los detalles de la computadora en particular que se use, aislándolo especialmente del procesamiento referido a las características internas de: memoria, discos, puertos y dispositivos de comunicaciones, impresoras, pantallas, teclados, etc. El software de sistema le gestiona al usuario y programador adecuadas interfaces de alto nivel, herramientas y utilidades de apoyo que permiten su mantenimiento. Incluye entre otros:

- Sistemas operativos
- Controladores de dispositivos
- Herramientas de diagnóstico
- Herramientas de Corrección y Optimización
- Servidores
- Utilidades

CONTENIDO DISCIPLINAR: TECNOLOGÍA DE LA INFORMACIÓN y SISTEMAS OPERATIVOS

Software de programación

- Es el conjunto de herramientas que permiten al programador desarrollar programas informáticos, usando diferentes alternativas y lenguajes de programación, de una manera práctica. Incluye entre otros:
- Editores de texto
- Compiladores
- Intérpretes
- Enlazadores
- Depuradores
- Entornos de Desarrollo Integrados (IDE): Agrupan las anteriores herramientas, usualmente en un entorno visual, de forma tal que el programador no necesite introducir múltiples comandos para compilar, interpretar, depurar, etc. Habitualmente cuentan con una avanzada interfaz gráfica de usuario (GUI).

Software de aplicación

- Es aquel que permite a los usuarios llevar a cabo una o varias tareas específicas, en cualquier campo de actividad susceptible de ser automatizado o asistido, con especial énfasis en los negocios. Incluye entre otros:

- Aplicaciones para Control de sistemas y automatización industrial (p.ej. Los conocidos sistemas de Supervisión, Control y Adquisición de Datos, SCADA)
- Aplicaciones ofimáticas (p.ej. aplicación de reconocimiento óptico de caracteres (OCR))
- Software educativo
- Software empresarial
- Bases de datos
- Telecomunicaciones (p.ej. Internet y toda su estructura lógica)
- Videojuegos
- Software médico
- Software de Cálculo Numérico y simbólico.
- Software de Diseño Asistido (CAD)
- Software de Control Numérico (CAM)

Software social

El software social no son propiamente aspectos de programación. Estas herramientas engloban correo electrónico, listas de correo electrónico, IRC, mensajería instantánea, bitácoras de red entre otros. Su empleo busca romper la separación y el aislamiento de los que participan en los programas a distancia y facilitar la construcción de conocimiento.

Características

1. Soporte de conversaciones entre individuos o grupos, que van desde los mensajes instantáneos en tiempo real hasta los espacios de colaboración en tiempo diferido.

2. Soporte para la retroalimentación que permita a un grupo conocer las contribuciones de los otros participantes y que lleva de forma implícita a la “reputación digital”.

3. Soporte a la red social para crear y conducir de forma explícita una expresión digital de las relaciones personales de un individuo y ayudarlo a adquirir nuevas relaciones.

Facilidades

CONTENIDO DISCIPLINAR: TECNOLOGÍA DE LA INFORMACIÓN y SISTEMAS OPERATIVOS

- Permite la comunicación entre grupos y entre personas.
- Compartir recursos.
- Indexación de la información –Referencias.
- Filtrado --Permitir la afiliación al sitio RSS.
- Herramientas que permiten modificación de los contenidos y sus nuevas. formulaciones- Creación de conocimiento.
- Herramientas de presencia.
- Ayuda mutua.

Existe consenso en que ya se puede hablar no sólo de las herramientas de comunicación de primera generación como el correo electrónico, foros de discusión y chat, sino también de toda una serie de servicios de segunda generación entre los que están los marcadores sociales.

Los blog son publicaciones sencillas sobre ideas, anécdotas y comentarios, que deseamos compartir con otros interlocutores. Hoy en día, los Blogs constituyen una forma eficaz de intercambio a través de una red. Promueve las relaciones entre los participantes.

Todas las aplicaciones y servicios Web que hoy ofrece INTERNET deben ser analizados por parte de los docentes antes de ser aplicados al proceso de enseñanza – aprendizaje.

El desarrollo del Web y su generalización están marcados por la calidad y madurez que han alcanzado los software de código abierto y la incorporación a las plataformas de nuevas versiones de software sociales.

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

1.2 Introducción a la Lógica en informática

La Informática es la ciencia que estudia el análisis y resolución de problemas utilizando computadoras.
La palabra ciencia se relaciona con una metodología fundamentada y racional para el estudio y resolución de los problemas. En este sentido la Informática se vincula especialmente con la Matemática.
Si se busca en el diccionario una definición en la palabra *problema* podrá hallarse alguna de las siguientes:

- Cuestión o proposición dudosa, que se trata de aclarar o resolver.
- Enunciado encaminado a averiguar el modo de obtener un resultado cuando se conocen ciertos datos.

La resolución de problemas mediante una computadora consiste en dar una adecuada formulación de pasos precisos a seguir.
Si se piensa en la forma en que una persona indica a otra como resolver un problema, se verá que habitualmente se utiliza un lenguaje común y corriente para realizar la explicación, quizá entremezclado con algunas palabras técnicas. Esto es un riesgo muy grande. Los que tienen cierta experiencia al respecto saben que es difícil transmitir el mensaje y por desgracia, con mucha frecuencia se malinterpretan las instrucciones y por lo tanto se ejecuta incorrectamente la solución obteniéndose errores.
Cuando de una computadora se trata, no pueden utilizarse indicaciones ambiguas. Ante cada orden resulta fundamental tener una única interpretación de lo que hay que realizar. Una máquina no posee la capacidad de decisión del ser humano para resolver situaciones no previstas. Si al dar una orden a la computadora se produce una situación no contemplada, será necesario abortar esa tarea y recomenzar todo el procedimiento nuevamente.
Además, para poder indicar a la computadora las órdenes que debe realizar es necesario previamente entender exactamente lo que se quiere hacer. Es fundamental conocer con qué información se cuenta y qué tipo de transformación se quiere hacer sobre ella.
A continuación se analizarán en forma general las distintas etapas que deben seguirse para poder llegar a resolver un problema utilizando una computadora como herramienta.

1.2 Etapas en la resolución de problemas con computadora

La resolución de problemas utilizando como herramienta una computadora no se resume únicamente en la escritura de un programa, sino que se trata de una tarea más compleja. El proceso abarca todos los aspectos que van desde interpretar las necesidades del usuario hasta verificar que la respuesta brindada es correcta. Las etapas son las siguientes:

Análisis del problema

En esta primera etapa, se analiza el problema en su contexto del mundo real. Deben obtenerse los requerimientos del usuario. El resultado de este análisis es un modelo preciso del ambiente del problema y del objetivo a resolver. Dos componentes importantes de este modelo son los datos a utilizar y las transformaciones de los mismos que llevan al objetivo.

Diseño de una solución

La resolución de un problema suele ser una tarea muy compleja para ser analizada como un todo. Una técnica de diseño en la resolución de problemas consiste en la identificación de las partes (subproblemas) que componen el problema y la manera en que se relacionan. Cada uno de estos subproblemas debe tener un objetivo específico, es decir, debe resolver una parte del problema original. La integración de las soluciones de los subproblemas es lo que permitirá obtener la solución buscada.

Especificación de algoritmos

La solución de cada subproblema debe ser especificada a través de un algoritmo. Esta etapa busca obtener la secuencia de pasos a seguir para resolver el problema. La elección del algoritmo adecuado es fundamental para garantizar la eficiencia de la solución.

Escritura de programas

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

Un algoritmo es una especificación simbólica que debe convertirse en un programa real sobre un lenguaje de programación concreto. A su vez, un programa escrito en un lenguaje de programación determinado (ej: Pascal, Ada, etc) es traducido automáticamente al lenguaje de máquina de la computadora que lo va a ejecutar. Esta traducción, denominada compilación, permite detectar y corregir los errores sintácticos que se cometan en la escritura del programa.

Verificación

Una vez que se tiene un programa escrito en un lenguaje de programación se debe verificar que su ejecución produce el resultado deseado, utilizando datos representativos del problema real. Sería deseable poder afirmar que el programa cumple con los objetivos para los cuales fue creado, más allá de los datos particulares de una ejecución. Sin embargo, en los casos reales es muy difícil realizar una verificación exhaustiva de todas las posibles condiciones de ejecución de un sistema de software. La facilidad de verificación y la depuración de errores de funcionamiento del programa conducen a una mejor calidad del sistema y es un objetivo central de la Ingeniería de Software.

En cada una de las etapas vistas se pueden detectar errores lo cual lleva a revisar aspectos de la solución analizados previamente.

Dada la sencillez de los problemas a resolver en este curso, la primera etapa correspondiente al análisis del problema, sólo se verá reflejada en la interpretación del enunciado a resolver. Sin embargo, a lo largo de la carrera se presentarán diferentes asignaturas que permitirán familiarizar al alumno con las técnicas necesarias para hacer frente a problemas de gran envergadura.

Con respecto a la segunda etapa, se pospondrá el análisis de este tema hasta el capítulo 5, ya que se comenzará a trabajar con problemas simples que no necesitan ser descompuestos en otros más elementales. Por lo tanto, a continuación se trabajará sobre el concepto de algoritmo como forma de especificación de soluciones concretas para la resolución de problemas con computadora.

1.3 Algoritmo

La palabra algoritmo deriva del nombre de un matemático árabe del siglo IX, llamado Al-Khuwarizmi, quien estaba interesado en resolver ciertos problemas de aritmética y describió varios métodos para resolverlos. Estos métodos fueron presentados como una lista de instrucciones específicas (como una receta de cocina) y su nombre es utilizado para referirse a dichos métodos.

Un algoritmo es, en forma intuitiva, una receta, un conjunto de instrucciones o de especificaciones sobre un proceso para hacer algo. Ese algo generalmente es la solución de un problema de algún tipo. Se espera que un algoritmo tenga varias **propiedades**. La primera es que un algoritmo **no debe ser ambiguo**, o sea, que si se trabaja dentro de cierto marco o contexto, cada instrucción del algoritmo debe significar sólo una cosa.

Se presentan a continuación algunos ejemplos:

Ejemplo 1.1:
Problema: Indique la manera de salar una masa.
Algoritmo 1: Ponerle algo de sal a la masa
Algoritmo 2: Agregarle una cucharadita de sal a la masa.
El algoritmo 1 presenta una solución ambigua al problema planteado.
El algoritmo 2 presenta una solución adecuada al problema.

Ejemplo 1.2:
Problema: Determinar la suma de todos los números enteros.
En este caso no se puede determinar un algoritmo para resolver este problema. Un algoritmo debe alcanzar la solución en un tiempo finito, situación que no se cumplirá en el ejemplo ya que los números enteros son infinitos.
Además de no ser ambiguo, un algoritmo debe detenerse. Se supone también que cuando se detiene, debe informar de alguna manera, su resultado. Es bastante factible escribir un conjunto de instrucciones que no incluyan una terminación y por lo tanto dicho conjunto de instrucciones no conformarían un algoritmo.

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

Ejemplo 1.3:
Problema: Volcar un montículo de arena en una zanja.
Algoritmo: Tome una pala. Mientras haya arena en el montículo cargue la pala con arena y vuélquela en la zanja. Dejar la pala.
Este algoritmo es muy simple y no ambiguo. Se está seguro que en algún momento parará, aunque no se sabe cuántas paladas se requerirán.
Resumiendo, un algoritmo puede definirse como una secuencia ordenada de pasos elementales, exenta de ambigüedades, que lleva a la solución de un problema dado en un tiempo finito.
Para comprender totalmente la definición anterior falta clarificar que se entiende por "paso elemental".

Ejemplo 1.4:
Escriba un algoritmo que permita preparar una tortilla de papas de tres huevos.
El enunciado anterior basta para que un cocinero experto lo resuelva sin mayor nivel de detalle, pero si este no es el caso, se deben describir los pasos necesarios para realizar la preparación. Esta descripción puede ser:
Mezclar papas cocidas, huevos y una pizca de sal en un recipiente
Freír
Esto podría resolver el problema, si el procesador o ejecutor del mismo no fuera una persona que da sus primeros pasos en tareas culinarias, ya que el nivel de detalle del algoritmo presupone muchas cosas.
Si este problema debe resolverlo una persona que no sabe cocinar, se debe detallar, cada uno de los pasos mencionados, pues estos no son lo bastante simples para un principiante.
De esta forma, el primer paso puede descomponerse en:
Pelar las papas
Cortarlas en cuadraditos
Cocinar las papas
Batir los huevos en un recipiente
Agregar las papas al recipiente y echar una pizca de sal al mismo
El segundo paso (freír) puede descomponerse en los siguientes tres: *Calentar el aceite en la sartén*
Verter el contenido del recipiente en la sartén
Dorar la tortilla de ambos lados
Nótese además que si la tortilla va a ser realizada por un niño, algunas tareas (por ejemplo batir los huevos) pueden necesitar una mejor especificación.
El ejemplo anterior sólo pretende mostrar que la lista de pasos elementales que compongan nuestro algoritmo depende de quién sea el encargado de ejecutarlo.
Si en particular, el problema va a ser resuelto utilizando una computadora, el conjunto de pasos elementales conocidos es muy reducido, lo que implica un alto grado de detalle para los algoritmos.
Se considera entonces como un paso elemental aquel que no puede volver a ser dividido en otros más simples.
De ahora en adelante se utilizará la palabra instrucción como sinónimo de paso elemental.
Un aspecto importante a discutir es el detalle que debe llevar el algoritmo. Esto no debe confundirse con el concepto anterior de paso elemental. En ocasiones, no se trata de descomponer una orden en acciones más simples sino que se busca analizar cuáles son las órdenes relevantes para el problema. Esto resulta difícil de cuantificar cuando las soluciones son expresadas en lenguaje natural. Analice el siguiente ejemplo:

Ejemplo 1.5:
Desarrolle un algoritmo que describa la manera en que Ud. se levanta todas las mañanas para ir al trabajo.
Salir de la cama
Quitarse el pijama
Ducharse
Vestirse
Desayunar
Arrancar el auto para ir al trabajo
Nótese que se ha llegado a la solución del problema en seis pasos, y no se resaltan aspectos como: colocarse un calzado después de salir de la cama, o abrir la llave de la ducha antes de ducharse. Estos aspectos han sido descartados, pues no tienen mayor trascendencia. En otras palabras se sobreentienden o se suponen. A nadie se le ocurriría ir a trabajar descalzo. En cambio existen aspectos que no pueden obviarse o suponerse porque el

algoritmo perdería lógica. El tercer paso, "vestirse", no puede ser omitido. Puede discutirse si requiere un mayor nivel de detalle o no, pero no puede ser eliminado del algoritmo.
Un buen desarrollador de algoritmos deberá reconocer esos aspectos importantes y tratar de simplificar su especificación de manera de seguir resolviendo el problema con la menor cantidad de órdenes posibles.

1.4 Pre y Postcondiciones de un algoritmo

Precondición es la información que se conoce como verdadera antes de comenzar el algoritmo.

En el ejemplo 1.1:
Problema: Indique la manera de salar una masa.
Algoritmo: Agregarle una cucharadita de sal a la masa.
Se supone que se dispone de todos los elementos para llevar a cabo esta tarea. Por lo tanto, como precondición puede afirmarse que se cuenta con la cucharita, la sal y la masa.

Postcondición es la información que se conoce como verdadera al concluir el algoritmo si se cumple adecuadamente el requerimiento pedido.

En el ejemplo 1.3:
Problema: Volcar un montículo de arena en una zanja.
Algoritmo: Tome una pala. Mientras haya arena en el montículo cargue la pala con arena y vuélquela en la zanja. Dejar la pala.

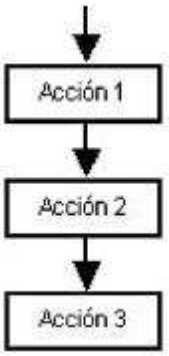
- ¿Cuáles serían las precondiciones y las postcondiciones del algoritmo?

La precondición es que se cuenta con la pala, la arena y está ubicado cerca de la zanja que debe llenar.
La postcondición es que el montículo quedó vació al terminar el algoritmo.

1.5 Elementos que componen un algoritmo

1.5.1 Secuencia de Acciones

Una secuencia de acciones está formada por una serie de instrucciones
que se ejecutan una a continuación de la otra.



Ejemplo 1.7: Escriba un algoritmo que permita cambiar una lámpara quemada.

Colocar la escalera debajo de la lámpara quemada
Tomar una lámpara nueva de la misma potencia que la anterior
Subir por la escalera con la nueva lámpara hasta alcanzar la lámpara a sustituir
Desenroscar la lámpara quemada
Enroscar la nueva lámpara hasta que quede apretada la nueva lámpara
Bajar de la escalera con lámpara quemada
Tirar la lámpara a la basura

Ejemplo 1.8: Escriba un algoritmo que permita a un robot subir 8 escalones

Levantar Pie Izquierdo
Subir un escalón
Levantar Pie Derecho
Subir un escalón
Levantar Pie Izquierdo
Subir un escalón
Levantar Pie Derecho
Subir un escalón
Levantar Pie Izquierdo
Subir un escalón
Levantar Pie Derecho
Subir un escalón
Levantar Pie Izquierdo
Subir escalón
Levantar Pie Derecho
Subir un escalón

Se denomina flujo de control de un algoritmo al orden en el cual deben ejecutarse los pasos individuales.

Hasta ahora se ha trabajado con flujo de control secuencial, es decir, la ejecución uno a uno de los pasos, desde el primero hasta el último.

Las estructuras de control son construcciones algorítmicas que alteran directamente el flujo de control secuencial del algoritmo.

Con ellas es posible seleccionar un determinado sentido de acción entre un par de alternativas específicas o repetir automáticamente un grupo de instrucciones.

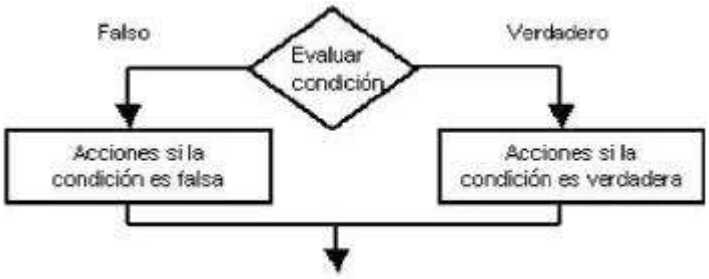
A continuación se presentan las estructuras de control necesarias para la resolución de problemas más complejos.

1.5.2 Selección

La escritura de soluciones a través de una secuencia de órdenes requiere conocer a priori las diferentes alternativas que se presentarán en la resolución del problema. Lamentablemente, es imposible contar con esta información antes de comenzar la ejecución de la secuencia de acciones.

Por ejemplo, que ocurriría si en el ejemplo 1.7 al querer sacar la lámpara quemada, el portalámparas se rompe. Esto implica que el resto de las acciones no podrán llevarse a cabo por lo que el algoritmo deberá ser interrumpido. Si se desea que esto no ocurra, el algoritmo deberá contemplar esta situación. Nótese que el estado del portalámparas es

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS



desconocido al iniciar el proceso y sólo es detectado al intentar sacar la lámpara quemada. Por lo que usar solamente la secuencia planteada es insuficiente para expresar esta solución.

A través de la selección se incorpora, a la especificación del algoritmo, la capacidad de decisión. De esta forma será posible seleccionar una de dos alternativas de acción posibles durante la ejecución del algoritmo.

Por lo tanto, el algoritmo debe considerar las dos alternativas, es decir, qué hacer en cada uno de los casos. La selección se notará de la siguiente forma:

si (condición)
acción o acciones a realizar si la condición es verdadera (1)
sino
acción acciones a realizar si la condición es falsa (2)

donde "condición" es una expresión que al ser evaluada puede tomar solamente uno de dos valores posibles: verdadero o falso.

El esquema anterior representa que en caso de que la condición a evaluar resulte verdadera se ejecutarán las acciones de (1) y NO se ejecutarán las de (2). En caso contrario, es decir, si la condición resulta ser falsa, solo se ejecutarán las acciones de (2).

En la figura 1.2 se grafica la selección utilizando un rombo para representar la decisión y un rectángulo para representar un bloque de acciones secuenciales.

Analice el siguiente ejemplo:

Ejemplo 1.9: Su amigo le ha pedido que le compre $1 de caramelos en el kiosco. De ser posible, prefiere que sean de menta pero si no hay, le da igual que sean de cualquier otro tipo. Escriba un algoritmo que represente esta situación.

Ir al kiosco
si(hay caramelos de menta)
Llevar caramelos de menta (1)

sino
Llevar de cualquier otro tipo
Pagar 1 peso

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

Los aspectos más importantes son:

(2)

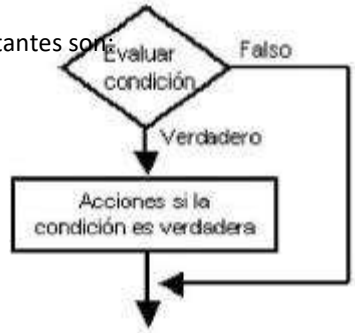


Figura 1.3: Estructura Si-Entonces

• No es posible saber si en el kiosco hay o no hay caramelos de menta ANTES de llegar al kiosco por lo que no puede utilizarse únicamente una secuencia de acciones para resolver este problema.

• La condición “hay caramelos de menta" sólo admite dos respuestas posibles: hay o no hay; es decir, verdadero o falso respectivamente.

• Si se ejecuta la instrucción marcada con (1), NO se ejecutará la acción (2) y viceversa.

• Independientemente del tipo de caramelos que haya comprado, siempre se pagará $1. Esta acción es independiente del tipo de caramelos que haya llevado.

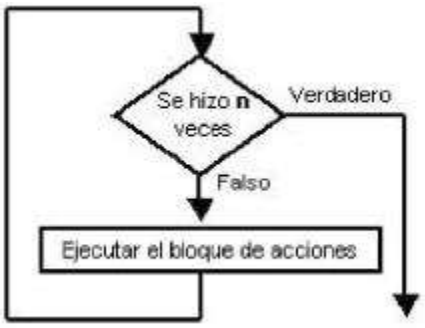
En algunos casos puede no haber una acción específica a realizar si la condición es falsa. En ese caso se utilizará la siguiente notación:

si (condición)
acción o acciones a realizar en caso de que la condición sea verdadera.

1.5.3 Repetición

Un componente esencial de los algoritmos es la repetición. La computadora, a diferencia de los humanos, posee una alta velocidad de procesamiento. A través de ella, es posible ejecutar, de manera repetitiva, algunos pasos elementales de un algoritmo.
Esto puede considerarse una extensión natural de la secuencia.

La repetición es la estructura de control que permite al algoritmo ejecutar un conjunto de instrucciones un número de veces fijo y conocido de antemano.



La notación a utilizar es la siguiente y se muestra en la figura 1.4:
repetir N
Acción o acciones a realizar N veces.
Se analizan a continuación algunos algoritmos que presentan repeticiones:
Ejemplo 1.11: Escriba un algoritmo que permita poner 4 litros de agua en un balde utilizando un vaso de 50 cc.
Al plantear una solución posible, se observa que hay dos pasos básicos: llenar el vaso con agua y vaciarlo en el

balde. Para completar los cuatro litros es necesario repetir estas dos operaciones ochenta veces. Suponga que se dispone de un vaso, un balde y una canilla para cargar el vaso con agua.

Tomar el vaso y el balde
repetir 80
Llenar el vaso de agua.
Vaciar el vaso en el balde.
Dejar el vaso y el balde.

Nótese que, la instrucción "Dejar el vaso y el balde" no pertenece a la repetición. Esto queda indicado por la sangría o indentación utilizada para cada instrucción. Por lo tanto, se repetirán 80 veces las instrucciones de "Llenar el vaso de agua" y "Vaciar el vaso en el balde".

Ejemplo 1.12: Escriba un algoritmo que permita a un robot subir 8 escalones.

repetir 4
LevantaPieIzquierdo
Subir un escalón.
LevantaPieDerecho
Subir un escalón

Este algoritmo realiza exactamente las mismas acciones que el algoritmo del ejemplo 1.8. Las ventajas de utilizar la repetición en lugar de la secuencia son: la reducción de la longitud del código y la facilidad de lectura.

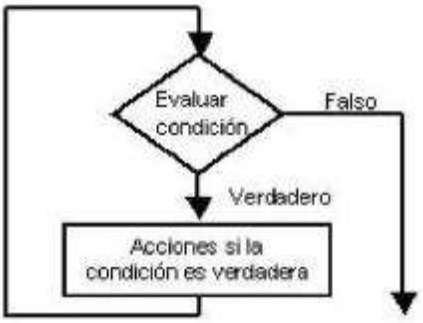
1.5.4 Iteración

Existen situaciones en las que se desconoce el número de veces que debe repetirse un conjunto de acciones. Por ejemplo, si se quiere llenar una zanja con arena utilizando una pala, será difícil indicar exactamente cuántas paladas de arena serán necesarias para realizar esta tarea. Sin embargo, se trata claramente de un proceso iterativo que consiste en cargar la pala y vaciarla en la zanja.
Por lo tanto, dentro de una iteración, además de una serie de pasos elementales que se repiten; es necesario contar con un mecanismo que lo detenga.

La iteración es una estructura de control que permite al algoritmo ejecutar en forma repetitiva un conjunto de acciones utilizando una condición para indicar su finalización.

El esquema iterativo es de la forma:

mientras *(condición)*
Acción o acciones a realizar en caso de que la condición sea verdad era.



Las acciones contenidas en la iteración serán ejecutadas mientras la condición sea verdadera. Es importante notar que, la primera vez, antes de ejecutar alguna de las acciones de la iteración, lo primero que se realiza es la evaluación de la condición. Sólo luego de comprobar que es verdadera se procede a ejecutar el conjunto de acciones pertenecientes al mientras.

Si inicialmente la condición resultara falsa, el contenido del mientras no se ejecutará ni siquiera una sola vez. Es importante que las acciones realizadas en el interior de la iteración modifiquen el valor de verdad de la condición a fin de garantizar que la iteración terminará en algún momento.

Analicemos el siguiente ejemplo:

Ejemplo 1.14: Escriba un algoritmo que permita volcar un montículo de arena en una zanja utilizando una pala.
Tomar la pala.
Ubicarse frente a la zanja.
mientras *(no esté vacío el montículo de arena)*
cargar la pala con arena
volcar la arena en la zanja
Dejar la pala.

La iteración indica que, mientras no se vacíe el montículo, se seguirá incorporando arena en la zanja. Cuando el montículo esté vacío, la condición será falsa y la iteración terminará. Es importante destacar, que si el montículo inicialmente estaba vacío, ninguna palada de arena será tomada del montículo ni incorporada a la zanja. Es decir, la condición se verifica ANTES de comenzar la iteración.

En este punto es apropiado hacerse la siguiente pregunta. ¿Qué sentido tiene introducer el concepto de iteración? Con toda seguridad, para los ejemplos antes mencionados no es necesario dicho concepto para establecer clara, simple o comprensiblemente las instrucciones del algoritmo.

Existe una razón bastante obvia para justificar esta estructura de control: es una realidad el hecho de que las computadoras requieren instrucciones detalladas y no ambiguas acerca de lo que deben hacer. Se debe, por lo tanto, dividir los algoritmos en pasos simples, de modo que las computadoras puedan efectuar sus cálculos. Si se quiere que algo sea realizado 80 veces, se le debe indicar que lo repita 80 veces. El empleo de las instrucciones de repetición, en este caso, permite hacer esto sin tener que escribir 80 líneas de instrucciones.

Por otro lado, el concepto de iteración es necesario para una mejor legibilidad o facilidad de lectura de los procesos algorítmicos. La iteración es un proceso fundamental en los algoritmos, y se debe ser capaz de pensar en términos de ciclos de iteración para poder construir los algoritmos.

1.6 Importancia de la indentación en las estructuras de control

Las instrucciones que pertenecen a una estructura de control deben tener una sangria mayor que la utilizada para escribir el comienzo de la estructura. De esta forma, podrá identificarse donde comienza y termina el conjunto de instrucciones involucradas en dicha estructura. A esta sangría se la denomina indentación.

Este concepto se aplica a las tres estructuras de control vistas previamente: selección, repetición e iteración.

El siguiente ejemplo muestra el uso de la indentación en la selección Ejemplo 1.15: Suponga que se planea una salida con amigos. La salida depende del clima: si llueve vos y tus amigos irán al cine a ver la película elegida, por el contrario si no llueve irán de pesca. Luego de realizar el paseo se juntarán a comentar la experiencia vivida. Escriba el algoritmo que resuelva esta situación.

Juntarse en una casa con el grupo de amigos
Mirar el estado del tiempo.
si *(llueve)* (1)
elegir película
ir al cine
sino

preparar el equipo de pesca
ir a la laguna a pescar
Volver a la casa a comentar sobre el paseo (2)
Como puede apreciarse, las acciones que deben ser realizadas cuando la condición es verdadera se encuentran desplazadas un poco más a la derecha que el resto de la estructura. Algo similar ocurre con las acciones a realizar cuando la condición es falsa. De esta forma puede diferenciarse lo que pertenece a la selección del resto de las instrucciones.

En el ejemplo anterior, la instrucción "Volver a la casa a comentar sobre el paseo" se realiza siempre sin importar si llovió o no. Esto se debe a que no pertenece a la selección. Esto queda de manifiesto al darle a las instrucciones (1) y (2) la misma indentación.

Ejemplo 1.16: Ud. desea ordenar una caja con 54 fotografías viejas de manera que todas queden al derecho; esto es, en la orientación correcta y la imagen boca arriba. Las fotografías ordenadas se irán guardando en el álbum familiar. Escriba el algoritmo que le permita resolver este problema.

Tomar la caja de fotos y un álbum vacío.
***repetir** 54*
Tomar una fotografía.
***si** (la foto está boca abajo)*
dar vuelta la foto
***si** (la foto no está en la orientación correcta)*
girar la foto para que quede en la orientación correcta
guardar la fotografía en el álbum
guardar el álbum

Según la indentación utilizada, la repetición contiene a la acción de "Tomar una fotografía", las dos selecciones y la instrucción "guardar la fotografía en el álbum". Las instrucciones "Tomar la caja de fotos y el álbum" y "Guardar el álbum" no pertenecen a la repetición.
Ejemplo 1.17: Ud. se dispone a tomar una taza de café con leche pero previamente debe endulzarlo utilizando azúcar en sobrecitos. Escriba un algoritmo que resuelva este problema.

Tomar la taza de café con leche.
Probar el café con leche
***mientras** (no esté lo suficientemente dulce el café)*
Tomar un sobre de azúcar.
Vaciar el contenido del sobre en la taza.
Mezclar para que el azúcar se disuelva.
Probar el café con leche
Tomar el café con leche.

Note que en este último ejemplo no se conoce de antemano la cantidad de sobrecitos de azúcar necesarios para endulzar el contenido de la taza. Además, la condición se evalúa antes de agregar el primer sobre. Según la indentación utilizada, la iteración incluye cuatro instrucciones. La acción "Tomar el café con leche" se ejecutará sólo cuando la iteración haya terminado, es decir, cuando la condición sea falsa.

1.7 Conclusiones

El uso de algoritmos permite expresar, de una forma clara, la manera en que un problema debe ser resuelto. Los elementos que lo componen son característicos de la resolución de problemas con computadora.
La ejercitación es la única herramienta para poder comprender y descubrir la verdadera potencialidad de las estructuras de control. Resulta fundamental alcanzar un total entendimiento del funcionamiento de estas estructuras para poder lograr expresar soluciones más complejas que los ejemplos aquí planteados.

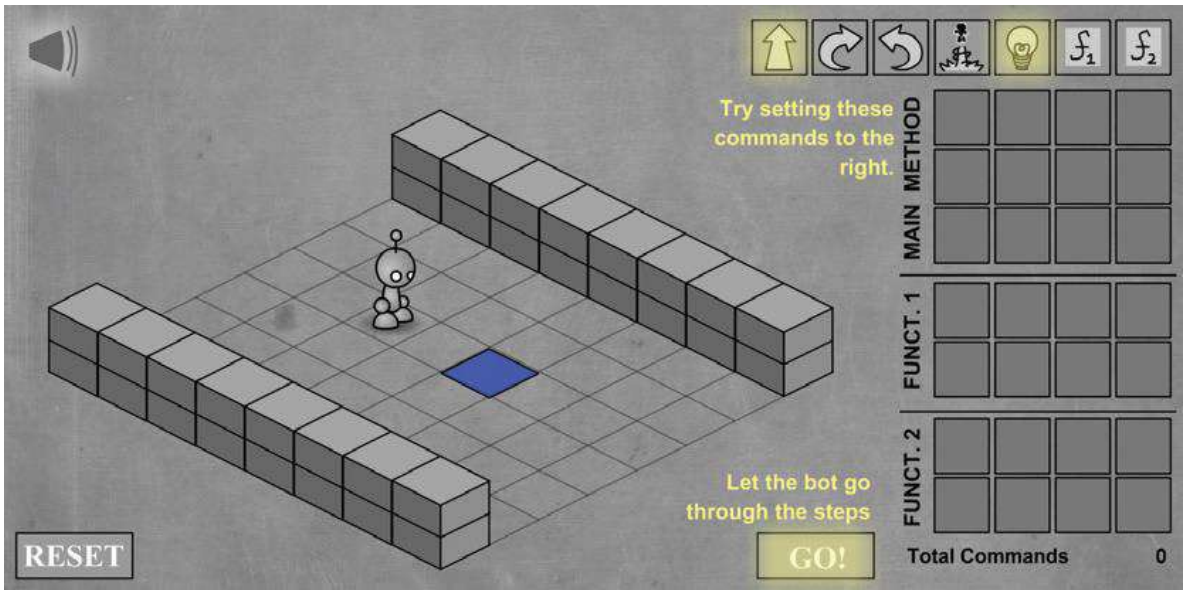
CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

INTRODUCCIÓN A LA PROGRAMACIÓN: LIGHTBOT Y SCRATCH

Sobre Lightbot

Lightbot es un juego de ingenio en el que el usuario debe programar para pasar de un nivel al siguiente. La dificultad va aumentando a medida que se pasa de nivel. Se utilizará esta herramienta para que los alumnos den sus primeros pasos en el desarrollo de programas que resuelven problemas. Deberán secuenciar acciones y, dado que el espacio para programar es limitado, abstraer subtareas a partir de tareas que se repiten.

Objetivo del juego: utilizando una serie de comandos, se debe lograr que el robot encienda todas las baldosas azules de la fábrica. Para empezar a jugar, hay que hacer clic en la opción New Game.



Los comandos primitivos que presenta el juego son los siguientes:

 **Avanzar** (hace que el robot avance un casillero, excepto si está frente a una pared o un desnivel, en cuyo caso permanece en el mismo lugar)

 **Girar a la izquierda**

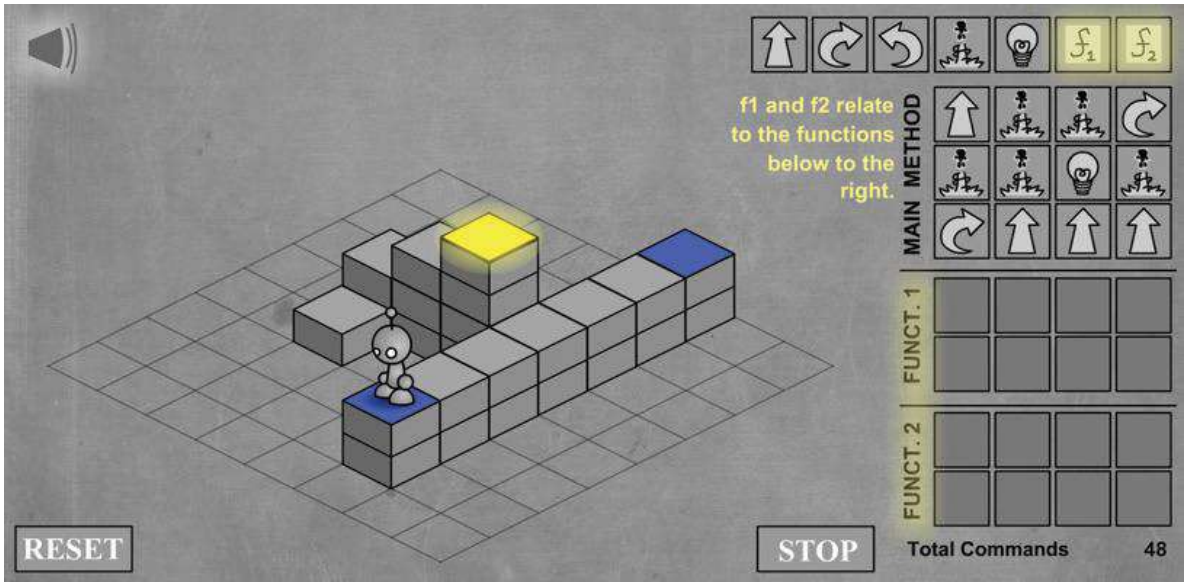
 **Girar a la derecha**

 **Saltar** (permite al robot saltar por encima de un solo bloque, o bien saltar de uno)

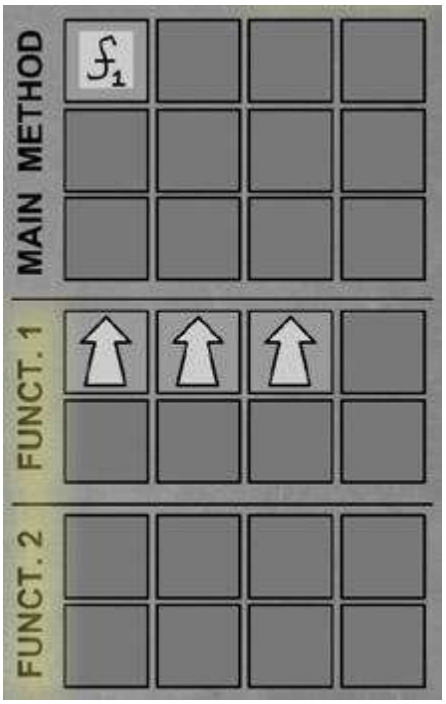
 **Encender luz** (si está apagada) o **Apagarla** (si está encendida)

Es relativamente sencillo avanzar hasta el nivel 5 inclusive; al llegar al nivel 6, es necesario comenzar a utilizar las subtareas que deja definir el juego (las grillas correspondientes a los procedimientos o funciones), porque no alcanzan las casillas disponibles en la primera grilla.

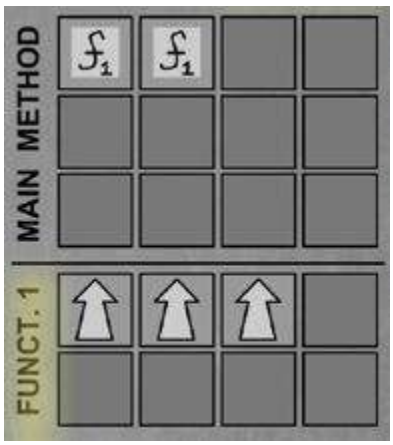
CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS



Mediante los comandos **F1** y **F2** se indica al robot que ejecute las nuevas tareas definidas. Así, por ejemplo, si el procedimiento o función 1 se define como *avanzar tres veces*, al colocar el comando en la primera grilla, se está indicando al robot que se mueva tres pasos hacia adelante:



Si se especifica dos veces **F1** , el robot se moverá seis veces hacia adelante. De este modo, en vez utilizar seis casilleros de la grilla, se utilizan solo dos ya que cada que cada **F1** vale por tres pasos:



Conclusión

Entre los aspectos importantes de esta aplicación, se encuentra el método de dividir un problema grande en partes más pequeñas, lo cual se conoce en programación como **división en subtareas**. La división en subtareas consiste en:

• Identificar pequeñas tareas fácilmente explicables (avanzar un paso, subir un peldaño, encender una luz) y asignarles un buen nombre para que se entienda qué representan.

• Combinar estas pequeñas tareas para definir el programa que resuelve el problema (los procedimientos, que en Lightbot se llaman *funciones*).

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

Sobre Scratch

Scratch es un lenguaje de programación diseñado para el aprendizaje de las herramientas básicas de programación. Los usuarios pueden crear sus propias historias interactivas, animaciones, juegos y música, y compartirlas en la web. Se puede acceder a él y utilizarlo gratuitamente ingresando en http://scratch.mit.edu En la propuesta que aquí se presenta, se trabajará a partir de actividades creadas ad hoc en Scratch. En cada actividad se plantea un problema específico, que los alumnos deberán resolver programando. Copiando en la barra del navegador la dirección que se indica en cada caso, se puede acceder a una versión on line de la actividad. Si bien todas las actividades de Scratch se diseñaron utilizando los personajes, objetos y fondos que provee el sitio, se debe tener presente que se pueden reemplazar por otros sin que ello implique una modificación de la actividad misma. En muchas ocasiones resultará conveniente cambiar algunos de esos elementos, en particular los personajes, eligiendo otros con los que están más familiarizados los alumnos con el fin de incentivar el aprendizaje.

En esta parte se trabajará con Scratch mediante un proyecto diseñado en ese lenguaje, que permite programar un autómata (representado en este caso por un gato). Las instrucciones u órdenes que debe realizar el autómata se organizan gráficamente en una serie de bloques. Cada bloque corresponde a una acción que debe realizar; cuando se los encastra verticalmente, forman una **secuencia de comandos**, es decir, una tarea que se hace en un orden establecido desde el primer bloque hasta el último (visualmente, desde arriba hacia abajo).

Luego de familiarizarse con el programa, acceder al ejercicio *El gato en la calle*. Si está en inglés, se puede cambiar el idioma de la interfaz, haciendo clic en el botón que se encuentra en el menú de la parte superior de la pantalla.

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

En esta primera parte, se sugiere trabajar solo con los bloques de acciones de la opción *Más bloques*, que se encuentra en la pestaña *Programas*. Haciendo clic en esa opción, se accede a la lista de acciones. Para que el autómata ejecute una acción, se arrastra el bloque correspondiente hacia el editor (el sector de la derecha) y se lo coloca debajo del bloque que dice *al recibir empezar.* Por ejemplo, en este caso, los alumnos seleccionarán el bloque *avanzar.*

A continuación, seleccionarán sucesivamente los bloques *saludar y volver*. El primer bloque irá debajo de *avanzar* y el segundo debajo de *saludar*. Así, se formará una secuencia de tres acciones para que las realice el gato.

Haciendo clic en la banderita verde que se observa en la parte superior de la pantalla se ejecutará el programa. El botón rojo sirve para detener el programa en cualquier momento. Para volver a ejecutar la secuencia, basta con hacer clic nuevamente en la banderita verde.

Hasta ahora, se han utilizado bloques que corresponden a acciones primitivas, es decir, acciones básicas preestablecidas, pero también pueden crearse nuevos bloques, es decir, nuevas acciones. A estas nuevas acciones se las llamará **procedimientos**. Por lo tanto, los procedimientos son nuevas acciones que el usuario le explica a la computadora cómo realizarlas.

Siguiendo con el ejemplo anterior, puede crearse un nuevo bloque que represente la actividad de dormirse y que pueda ejercitarse directamente. Para ello, se presiona, dentro de la opción *Más bloques*, el botón *Crear un bloque*. Al hacerlo, se abre una ventana, con un bloque en blanco, en el cual se escribe el nombre del nuevo bloque (en este caso, *dormirse*).

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

Repetición simple

Con esta secuencia, se empieza a trabajar en procesos más específicos, que los alumnos irán explorando sobre la base de lo ya aprendido. En este caso, la propuesta consiste en indagar, a través de una actividad de Scratch, en los procesos informáticos implicados en la repetición de tareas.

Los alumnos ingresarán a la actividad de Scratch *No me canso de saltar*, donde se presenta el siguiente escenario.

El objetivo es hacer que el gato salte 30 veces para llegar a la última piedra. Para eso, en la categoría *Más bloques,* solo se da la siguiente primitiva: **saltar**

Cuando se ejecuta una vez *saltar,* el gato pasa de una piedra a otra y avisa cuántos saltos le faltan para cumplir el objetivo:

Probablemente, la solución que propongan consista en repetir 30 veces el comando *saltar*: Además de poco práctica, esta solución presenta el inconveniente de que puede hacer que sea difícil visualizar la manera de modificar la secuencia. Se vuelve a presentar un problema similar al planteado en la secuencia didáctica anterior, cuando, con el fin de ahorrar las casillas disponibles, se recurrió a la definición de diversos procedimientos o funciones, cada uno formado por dos o más subtareas. Puede suceder que algunos alumnos, teniendo esto presente, pregunten si existe algún tipo de comando que permita *economizar* la cantidad de bloques, de manera que la realización de los saltos pueda indicarse de forma menos engorrosa.

Una vez delimitado el problema, se presenta el bloque *repetir*, ubicado en la categoría *Control*:

Tal como se hace con cualquier otro bloque que se desea utilizar, se lo arrastra hasta el editor. Luego, en la casilla de arriba se indica la cantidad de veces que se desea repetir una acción. En el hueco de abajo, se ubica la secuencia de comandos (en este caso, la acción que ejecutará repetidamente el gato)

Ejercitaciones

Se propone llevar a cabo las siguientes actividades para practicar lo desarrollado hasta el momento:

- *Lightbot en Scratch*
- *El recolector de estrellas*
- *María, la come sandías*
- *Alimentando a los peces*

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

- *Instalando juegos*
- *La gran aventura del mar encantado*
- *Reparando la nave*

Escenarios cambiantes

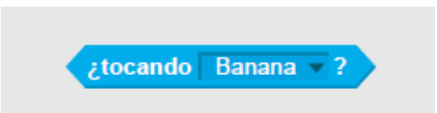
En esta secuencia se reflexionará sobre el uso de alternativas condicionales en programación, a partir de actividades en las que no se utilizarán computadoras.

Alternativas condicionales en Scratch

En esta secuencia se trabajará con una serie de actividades de Scratch que presentan un escenario cambiante y cuya resolución implica el uso de alternativas condicionales.
Se comenzará con el proyecto de Scratch *El mono y las bananas*. Al ingresar y hacer clic varias veces en , se podrán observar aleatoriamente los siguientes cambios de escenario:

El objetivo es simple: lograr que el mono coma una banana sólo cuando hay una banana. Para ello, se cuenta con las siguientes acciones primitivas: **avanzar y comer banana**
Para poder resolver la actividad, es necesario utilizar un bloque nuevo (ya presentado en la actividad anterior): **si… entonces.** Este bloque, que se encuentra en la categoría Control, permite, como se recordará, ejecutar un comando solo en el caso de cumplirse cierta condición.
Sin embargo, antes de intentar ver cómo utilizar este bloque, conviene dividir el problema, es decir, establecer la secuencia de acciones adecuada para el objetivo. Ello implica definir al menos un procedimiento, al que se puede llamar *comer banana si hay*, que resuelva el problema de comer una banana sólo en el caso de que esta exista.
Ahora bien, *¿cómo definir comer banana si hay?* Puesto que de lo que se trata es de que el mono coma la banana si existe, es necesario darle alguna indicación para que *sepa* si la fruta está presente.

Para que el mono reconozca si la fruta está presente y la coma, se presentará el bloque *¿tocando…?* ubicado en la categoría *Sensores.*
Mediante este bloque, es posible hacer que el mono verifique si está tocando la fruta. Para ello, se hace clic exactamente sobre el casillero con la punta de flecha que se encuentra junto a tocando y se selecciona la banana:



Esta primera parte de la instrucción se lee: “si tocando banana, entonces…”. Para completarla, es necesario elegir el comando que se ejecutará en el caso de que se cumpla esa condición.

CONTENIDO DISCIPLINAR: LÓGICA Y ESTRUCTURA DE DATOS

Ejercitaciones

Se propone llevar a cabo las siguientes actividades para practicar lo desarrollado hasta el momento:

- Laberinto corto
- Tres naranjas
- Lightbot recargado
- Laberinto largo

CONTENIDO DISCIPLINAR: INGENIERíA DE SOFTWARE

CAPÍTULO

1

EL SOFTWARE Y LA INGENIERÍA DE SOFTWARE

CONCEPTOS CLAVE

Tenía la apariencia clásica de un alto ejecutivo de una compañía importante de software —a la mitad de los 40, con las sienes comenzando a encanecer, esbelto y atlético, con ojos que penetraban al observador mientras hablaba—. Pero lo que dijo me dejó anonadado. "El software ha *muerto*".

Pestañeé con sorpresa y sonreí. "Bromeas, ¿verdad? El mundo es dirigido con software y tu empresa se ha beneficiado mucho de ello. ¡No ha muerto! Está vivo y en desarrollo."

Movió su cabeza de manera enfática. "No, está muerto... al menos como lo conocimos."

Me apoyé en el escritorio. "Continúa."

Habló al tiempo que golpeaba en la mesa con énfasis. "El concepto antiguo del software —lo compras, lo posees y tu trabajo consiste en administrarlo— está llegando a su fin. Hoy día, con Web 2.0 y la computación ubicua cada vez más fuerte, vamos a ver una generación de software por completo diferente. Se distribuirá por internet y se verá exactamente como si estuviera instalado en el equipo de cómputo de cada usuario... pero se encontrará en un servidor remoto."

Tuve que estar de acuerdo. "Entonces, tu vida será mucho más sencilla. Tus muchachos no tendrán que preocuparse por las cinco diferentes versiones de la misma App que utilizan decenas de miles de usuarios."

Sonrió. "Absolutamente. Sólo la versión más reciente estará en nuestros servidores. Cuando hagamos un cambio o corrección, actualizaremos funcionalidad y contenido a cada usuario. Todos lo tendrán en forma instantánea..."

Hice una mueca. "Pero si cometes un error, todos lo tendrán también instantáneamente".

Él se rió entre dientes. "Es verdad, por eso estamos redoblando nuestros esfuerzos para hacer una ingeniería de software aún mejor. El problema es que tenemos que hacerlo 'rápido' porque el mercado se ha acelerado en cada área de aplicación."

UNA MIRADA RÁPIDA

¿Qué es? El software de computadora es el producto que construyen los programadores profesionales y al que después le dan mantenimiento durante un largo tiempo. Incluye programas que se ejecutan en una computadora de cualquier tamaño y arquitectura, contenido que se presenta a medida de que se ejecutan los programas de cómputo e información descriptiva tanto en una copia dura como en formatos virtuales que engloban virtualmente a cualesquiera medios electrónicos. La ingeniería de software está formada por un proceso, un conjunto de métodos (prácticas) y un arreglo de herramientas que permite a los profesionales elaborar software de cómputo de alta calidad.

¿Quién lo hace? Los ingenieros de software elaboran y dan mantenimiento al software, y virtualmente cada persona lo emplea en el mundo industrializado, ya sea en forma directa o indirecta.

¿Por qué es importante? El software es importante porque afecta a casi todos los aspectos de nuestras vidas y ha invadido nuestro comercio, cultura y actividades cotidianas. La ingeniería de software es importante porque nos permite construir sistemas complejos en un tiempo razonable y con alta calidad.

¿Cuáles son los pasos? El software de computadora se construye del mismo modo que cualquier producto exitoso, con la aplicación de un proceso ágil y adaptable para obtener un resultado de mucha calidad, que satisfaga las necesidades de las personas que usarán el producto. En estos pasos se aplica el enfoque de la ingeniería de software.

¿Cuál es el producto final? Desde el punto de vista de un ingeniero de software, el producto final es el conjunto de programas, contenido (datos) y otros productos terminados que constituyen el software de computadora. Pero desde la perspectiva del usuario, el producto final es la información resultante que de algún modo hace mejor al mundo en el que vive.

¿Cómo me aseguro de que lo hice bien? Lea el resto de este libro, seleccione aquellas ideas que sean aplicables al software que usted hace y aplíquelas a su trabajo.

Me recargué en la espalda y coloqué mis manos en mi nuca. "Ya sabes lo que se dice... puedes tenerlo rápido o bien hecho o barato. Escoge dos de estas características..."

"Elijo rápido y bien hecho", dijo mientras comenzaba a levantarse.

También me incorporé. "Entonces realmente necesitas ingeniería de software."

"Ya lo sé", dijo mientras salía. "El problema es que tenemos que llegar a convencer a otra generación más de técnicos de que así es..."

¿Está muerto *realmente* el software? Si lo estuviera, usted no estaría leyendo este libro...

Cita:

"Las ideas y los descubrimientos tecnológicos son los motores que impulsan el crecimiento económico."

Wall Street Journal

El software de computadora sigue siendo la tecnología más importante en la escena mundial. Y también es un ejemplo magnífico de la ley de las consecuencias inesperadas. Hace 50 años, nadie hubiera podido predecir que el software se convertiría en una tecnología indispensable para los negocios, ciencias e ingeniería, ni que permitiría la creación de tecnologías nuevas (por ejemplo, ingeniería genética y nanotecnología), la ampliación de tecnologías ya existentes (telecomunicaciones) y el cambio radical de tecnologías antiguas (la industria de la impresión); tampoco que el software sería la fuerza que impulsaría la revolución de las computadoras personales, que productos de software empacados se comprarían en los supermercados, que el software evolucionaría poco a poco de un producto a un servicio cuando compañías de software "sobre pedido" proporcionaran funcionalidad justo a tiempo a través de un navegador web, que una compañía de software sería más grande y tendría más influencia que casi todas las empresas de la era industrial, que una vasta red llamada internet sería operada con software y evolucionaría y cambiaría todo, desde la investigación en bibliotecas y la compra de productos para el consumidor hasta el discurso político y los hábitos de encuentro de los adultos jóvenes (y no tan jóvenes).

Nadie pudo prever que habría software incrustado en sistemas de toda clase: de transporte, médicos, de telecomunicaciones, militares, industriales, de entretenimiento, en máquinas de oficina... la lista es casi infinita. Y si usted cree en la ley de las consecuencias inesperadas, hay muchos efectos que aún no podemos predecir.

Nadie podía anticipar que millones de programas de computadora tendrían que ser corregidos, adaptados y mejorados a medida que transcurriera el tiempo. Ni que la carga de ejecutar estas actividades de "mantenimiento" absorbería más personas y recursos que todo el trabajo aplicado a la creación de software nuevo.

Conforme ha aumentado la importancia del software, la comunidad de programadores ha tratado continuamente de desarrollar tecnologías que hagan más fácil, rápida y barata la elaboración de programas de cómputo de alta calidad. Algunas de estas tecnologías se dirigen a un dominio específico de aplicaciones (por ejemplo, diseño e implantación de un sitio web), otras se centran en un dominio tecnológico (sistemas orientados a objetos o programación orientada a aspectos), otros más tienen una base amplia (sistemas operativos, como Linux). Sin embargo, todavía falta por desarrollarse una tecnología de software que haga todo esto, y hay pocas probabilidades de que surja una en el futuro. A pesar de ello, las personas basan sus trabajos, confort, seguridad, diversiones, decisiones y sus propias vidas en software de computadora. Más vale que esté bien hecho.

Este libro presenta una estructura que puede ser utilizada por aquellos que hacen software de cómputo —personas que deben hacerlo bien—. La estructura incluye un proceso, un conjunto de métodos y unas herramientas que llamamos *ingeniería de software*.

1.1 La naturaleza del software

En la actualidad, el software tiene un papel dual. Es un producto y al mismo tiempo es el vehículo para entregar un producto. En su forma de producto, brinda el potencial de cómputo incorporado en el hardware de cómputo o, con más amplitud, en una red de computadoras a las

PUNTO CLAVE

El software es tanto un producto como un vehículo para entregar un producto.

que se accede por medio de un hardware local. Ya sea que resida en un teléfono móvil u opere en el interior de una computadora central, el software es un transformador de información —produce, administra, adquiere, modifica, despliega o transmite información que puede ser tan simple como un solo bit o tan compleja como una presentación con multimedios generada a partir de datos obtenidos de decenas de fuentes independientes—. Como vehículo utilizado para distribuir el producto, el software actúa como la base para el control de la computadora (sistemas operativos), para la comunicación de información (redes) y para la creación y control de otros programas (herramientas y ambientes de software).

Cita:

"El software es un lugar donde se siembran sueños y se cosechan pesadillas, una ciénega abstracta y mística en la que terribles demonios luchan contra panaceas mágicas, un mundo de hombres lobo y balas de plata."

Brad J. Cox

El software distribuye el producto más importante de nuestro tiempo: *información*. Transforma los datos personales (por ejemplo, las transacciones financieras de un individuo) de modo que puedan ser más útiles en un contexto local, administra la información de negocios para mejorar la competitividad, provee una vía para las redes mundiales de información (la internet) y brinda los medios para obtener información en todas sus formas.

En el último medio siglo, el papel del software de cómputo ha sufrido un cambio significativo. Las notables mejoras en el funcionamiento del hardware, los profundos cambios en las arquitecturas de computadora, el gran incremento en la memoria y capacidad de almacenamiento, y una amplia variedad de opciones de entradas y salidas exóticas han propiciado la existencia de sistemas basados en computadora más sofisticados y complejos. Cuando un sistema tiene éxito, la sofisticación y complejidad producen resultados deslumbrantes, pero también plantean problemas enormes para aquellos que deben construir sistemas complejos.

En la actualidad, la enorme industria del software se ha convertido en un factor dominante en las economías del mundo industrializado. Equipos de especialistas de software, cada uno centrado en una parte de la tecnología que se requiere para llegar a una aplicación compleja, han reemplazado al programador solitario de los primeros tiempos. A pesar de ello, las preguntas que se hacía aquel programador son las mismas que surgen cuando se construyen sistemas modernos basados en computadora:[1]

- ¿Por qué se requiere tanto tiempo para terminar el software?
- ¿Por qué son tan altos los costos de desarrollo?
- ¿Por qué no podemos detectar todos los errores antes de entregar el software a nuestros clientes?
- ¿Por qué dedicamos tanto tiempo y esfuerzo a mantener los programas existentes?
- ¿Por qué seguimos con dificultades para medir el avance mientras se desarrolla y mantiene el software?

Éstas y muchas otras preguntas, denotan la preocupación sobre el software y la manera en que se desarrolla, preocupación que ha llevado a la adopción de la práctica de la ingeniería del software.

1.1.1 Definición de software

En la actualidad, la mayoría de profesionales y muchos usuarios tienen la fuerte sensación de que entienden el software. Pero, ¿es así?

? ¿Cómo se define *software*?

La descripción que daría un libro de texto sobre software sería más o menos así:

> El software es: 1) instrucciones (programas de cómputo) que cuando se ejecutan proporcionan las características, función y desempeño buscados; 2) estructuras de datos que permiten que los progra-

1 En un excelente libro de ensayos sobre el negocio del software, Tom DeMarco [DeM95] defiende el punto de vista contrario. Dice: "En lugar de preguntar por qué el software cuesta tanto, necesitamos comenzar a preguntar: ¿Qué hemos hecho para hacer posible que el software actual cueste tan poco? La respuesta a esa pregunta nos ayudará a continuar el extraordinario nivel de logro que siempre ha distinguido a la industria del software."

mas manipulen en forma adecuada la información, y 3) información descriptiva tanto en papel como en formas virtuales que describen la operación y uso de los programas.

No hay duda de que podrían darse definiciones más completas.

Pero es probable que una definición más formal no mejore de manera apreciable nuestra comprensión. Para asimilar lo anterior, es importante examinar las características del software que lo hacen diferente de otros objetos que construyen los seres humanos. El software es elemento de un sistema lógico y no de uno físico. Por tanto, tiene características que difieren considerablemente de las del hardware:

1. *El software se desarrolla o modifica con intelecto; no se manufactura en el sentido clásico.*

PUNTO CLAVE
El software se modifica con intelecto, no se manufactura.

Aunque hay algunas similitudes entre el desarrollo de software y la fabricación de hardware, las dos actividades son diferentes en lo fundamental. En ambas, la alta calidad se logra a través de un buen diseño, pero la fase de manufactura del hardware introduce problemas de calidad que no existen (o que se corrigen con facilidad) en el software. Ambas actividades dependen de personas, pero la relación entre los individuos dedicados y el trabajo logrado es diferente por completo (véase el capítulo 24). Las dos actividades requieren la construcción de un "producto", pero los enfoques son distintos. Los costos del software se concentran en la ingeniería. Esto significa que los proyectos de software no pueden administrarse como si fueran proyectos de manufactura.

2. *El software no se "desgasta".*

PUNTO CLAVE
El software no se desgasta, pero sí se deteriora.

La figura 1.1 ilustra la tasa de falla del hardware como función del tiempo. La relación, que es frecuente llamar "curva de tina", indica que el hardware presenta una tasa de fallas relativamente elevada en una etapa temprana de su vida (fallas que con frecuencia son atribuibles a defectos de diseño o manufactura); los defectos se corrigen y la tasa de fallas se abate a un nivel estable (muy bajo, por fortuna) durante cierto tiempo. No obstante, conforme pasa el tiempo, la tasa de fallas aumenta de nuevo a medida que los componentes del hardware resienten los efectos acumulativos de suciedad, vibración, abuso, temperaturas extremas y muchos otros inconvenientes ambientales. En pocas palabras, el hardware comienza a *desgastarse*.

El software no es susceptible a los problemas ambientales que hacen que el hardware se desgaste. Por tanto, en teoría, la curva de la tasa de fallas adopta la forma de la "curva idealizada" que se aprecia en la figura 1.2. Los defectos ocultos ocasionarán ta-

FIGURA 1.1
Curva de fallas del hardware

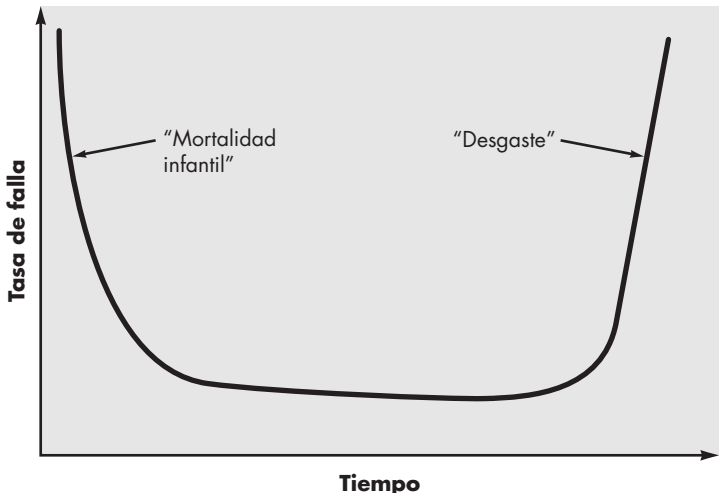
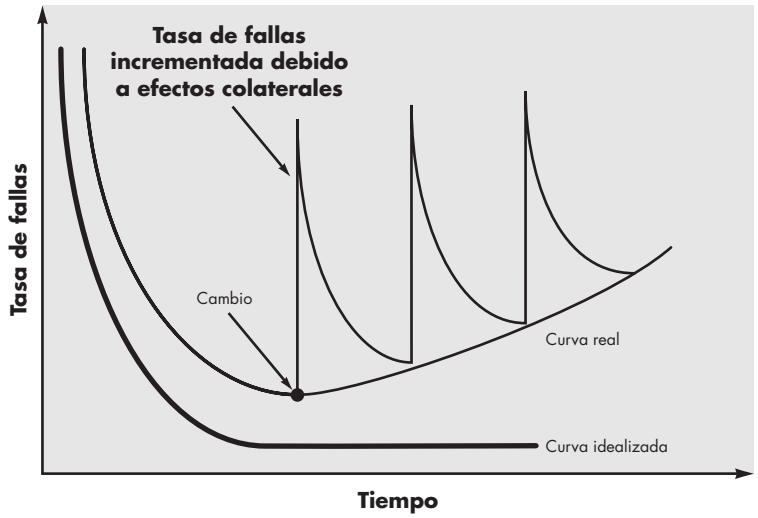


FIGURA 1.2
Curvas de falla del software



Si quiere reducir el deterioro del software, tendrá que mejorar su diseño (capítulos 8 a 13).

Los métodos de la ingeniería de software llevan a reducir la magnitud de los picos y de la pendiente de la curva real en la figura 1.2.

sas elevadas de fallas al comienzo de la vida de un programa. Sin embargo, éstas se corrigen y la curva se aplana, como se indica. La curva idealizada es una gran simplificación de los modelos reales de las fallas del software. Aun así, la implicación está clara: el software no se desgasta, ¡pero sí se *deteriora!*

Esta contradicción aparente se entiende mejor si se considera la curva real en la figura 1.2. Durante su vida,[2] el software sufrirá cambios. Es probable que cuando éstos se realicen, se introduzcan errores que ocasionen que la curva de tasa de fallas tenga aumentos súbitos, como se ilustra en la "curva real" (véase la figura 1.2). Antes de que la curva vuelva a su tasa de fallas original de estado estable, surge la solicitud de otro cambio que hace que la curva se dispare otra vez. Poco a poco, el nivel mínimo de la tasa de fallas comienza a aumentar: el software se está deteriorando como consecuencia del cambio.

Otro aspecto del desgaste ilustra la diferencia entre el hardware y el software. Cuando un componente del hardware se desgasta es sustituido por una refacción. En cambio, no hay refacciones para el software. Cada falla de éste indica un error en el diseño o en el proceso que tradujo el diseño a código ejecutable por la máquina. Entonces, las tareas de mantenimiento del software, que incluyen la satisfacción de peticiones de cambios, involucran una complejidad considerablemente mayor que el mantenimiento del hardware.

Cita:

"Las ideas son los ladrillos con los que se construyen las ideas."

Jason Zebehazy

3. *Aunque la industria se mueve hacia la construcción basada en componentes, la mayor parte del software se construye para un uso individualizado*.

A medida que evoluciona una disciplina de ingeniería, se crea un conjunto de componentes estandarizados para el diseño. Los tornillos estándar y los circuitos integrados preconstruidos son sólo dos de los miles de componentes estándar que utilizan los ingenieros mecánicos y eléctricos conforme diseñan nuevos sistemas. Los componentes reutilizables han sido creados para que el ingeniero pueda concentrarse en los elementos verdaderamente innovadores de un diseño; es decir, en las partes de éste que representan algo nuevo. En el mundo del hardware, volver a usar componentes es una parte

2 En realidad, los distintos participantes solicitan cambios desde el momento en que comienza el desarrollo y mucho antes de que se disponga de la primera versión.

natural del proceso de ingeniería. En el del software, es algo que apenas ha empezado a hacerse a gran escala.

Un componente de software debe diseñarse e implementarse de modo que pueda volverse a usar en muchos programas diferentes. Los modernos componentes reutilizables incorporan tanto los datos como el procesamiento que se les aplica, lo que permite que el ingeniero de software cree nuevas aplicaciones a partir de partes susceptibles de volverse a usar.[3] Por ejemplo, las actuales interfaces interactivas de usuario se construyen con componentes reutilizables que permiten la creación de ventanas gráficas, menús desplegables y una amplia variedad de mecanismos de interacción. Las estructuras de datos y el detalle de procesamiento que se requieren para construir la interfaz están contenidos en una librería de componentes reusables para tal fin.

1.1.2 Dominios de aplicación del software

Actualmente, hay siete grandes categorías de software de computadora que plantean retos continuos a los ingenieros de software:

Software de sistemas: conjunto de programas escritos para dar servicio a otros programas. Determinado software de sistemas (por ejemplo, compiladores, editores y herramientas para administrar archivos) procesa estructuras de información complejas pero deterministas.[4] Otras aplicaciones de sistemas (por ejemplo, componentes de sistemas operativos, manejadores, software de redes, procesadores de telecomunicaciones) procesan sobre todo datos indeterminados. En cualquier caso, el área de software de sistemas se caracteriza por: gran interacción con el hardware de la computadora, uso intensivo por parte de usuarios múltiples, operación concurrente que requiere la secuenciación, recursos compartidos y administración de un proceso sofisticado, estructuras complejas de datos e interfaces externas múltiples.

WebRef

En la dirección **shareware.cnet.com** se encuentra una de las librerías más completas de software compartido y libre.

Software de aplicación: programas aislados que resuelven una necesidad específica de negocios. Las aplicaciones en esta área procesan datos comerciales o técnicos en una forma que facilita las operaciones de negocios o la toma de decisiones administrativas o técnicas. Además de las aplicaciones convencionales de procesamiento de datos, el software de aplicación se usa para controlar funciones de negocios en tiempo real (por ejemplo, procesamiento de transacciones en punto de venta, control de procesos de manufactura en tiempo real).

Software de ingeniería y ciencias: se ha caracterizado por algoritmos "devoradores de números". Las aplicaciones van de la astronomía a la vulcanología, del análisis de tensiones en automóviles a la dinámica orbital del transbordador espacial, y de la biología molecular a la manufactura automatizada. Sin embargo, las aplicaciones modernas dentro del área de la ingeniería y las ciencias están abandonando los algoritmos numéricos convencionales. El diseño asistido por computadora, la simulación de sistemas y otras aplicaciones interactivas, han comenzado a hacerse en tiempo real e incluso han tomado características del software de sistemas.

Software incrustado: reside dentro de un producto o sistema y se usa para implementar y controlar características y funciones para el usuario final y para el sistema en sí. El software incrustado ejecuta funciones limitadas y particulares (por ejemplo, control del tablero de un horno de microondas) o provee una capacidad significativa de funcionamiento y control

3 El desarrollo basado en componentes se estudia en el capítulo 10.

4 El software es *determinista* si es posible predecir el orden y momento de las entradas, el procesamiento y las salidas. El software es *no determinista* si no pueden predecirse el orden y momento en que ocurren éstos.

(funciones digitales en un automóvil, como el control del combustible, del tablero de control y de los sistemas de frenado).

Software de línea de productos: es diseñado para proporcionar una capacidad específica para uso de muchos consumidores diferentes. El software de línea de productos se centra en algún mercado limitado y particular (por ejemplo, control del inventario de productos) o se dirige a mercados masivos de consumidores (procesamiento de textos, hojas de cálculo, gráficas por computadora, multimedios, entretenimiento, administración de base de datos y aplicaciones para finanzas personales o de negocios).

Aplicaciones web: llamadas "*webapps*", esta categoría de software centrado en redes agrupa una amplia gama de aplicaciones. En su forma más sencilla, las *webapps* son poco más que un conjunto de archivos de hipertexto vinculados que presentan información con uso de texto y gráficas limitadas. Sin embargo, desde que surgió Web 2.0, las *webapps* están evolucionando hacia ambientes de cómputo sofisticados que no sólo proveen características aisladas, funciones de cómputo y contenido para el usuario final, sino que también están integradas con bases de datos corporativas y aplicaciones de negocios.

Cita:

"No hay computadora que tenga sentido común."

Marvin Minsky

Software de inteligencia artificial: hace uso de algoritmos no numéricos para resolver problemas complejos que no son fáciles de tratar computacionalmente o con el análisis directo. Las aplicaciones en esta área incluyen robótica, sistemas expertos, reconocimiento de patrones (imagen y voz), redes neurales artificiales, demostración de teoremas y juegos.

Son millones de ingenieros de software en todo el mundo los que trabajan duro en proyectos de software en una o más de estas categorías. En ciertos casos se elaboran sistemas nuevos, pero en muchos otros se corrigen, adaptan y mejoran aplicaciones ya existentes. No es raro que una joven ingeniera de software trabaje en un programa de mayor edad que la de ella… Las generaciones pasadas de los trabajadores del software dejaron un legado en cada una de las categorías mencionadas. Por fortuna, la herencia que dejará la actual generación aligerará la carga de los futuros ingenieros de software. Aun así, nuevos desafíos (capítulo 31) han aparecido en el horizonte.

Computación en un mundo abierto: el rápido crecimiento de las redes inalámbricas quizá lleve pronto a la computación verdaderamente ubicua y distribuida. El reto para los ingenieros de software será desarrollar software de sistemas y aplicación que permita a dispositivos móviles, computadoras personales y sistemas empresariales comunicarse a través de redes enormes.

Construcción de redes: la red mundial (World Wide Web) se está convirtiendo con rapidez tanto en un motor de computación como en un proveedor de contenido. El desafío para los ingenieros de software es hacer arquitecturas sencillas (por ejemplo, planeación financiera personal y aplicaciones sofisticadas que proporcionen un beneficio a mercados objetivo de usuarios finales en todo el mundo).

Fuente abierta: tendencia creciente que da como resultado la distribución de código fuente para aplicaciones de sistemas (por ejemplo, sistemas operativos, bases de datos y ambientes de desarrollo) de modo que mucha gente pueda contribuir a su desarrollo. El desafío para los ingenieros de software es elaborar código fuente que sea autodescriptivo, y también, lo que es más importante, desarrollar técnicas que permitirán tanto a los consumidores como a los desarrolladores saber cuáles son los cambios hechos y cómo se manifiestan dentro del software.

Cita:

"No siempre puedes predecir, pero siempre puedes prepararte."

Anónimo

Es indudable que cada uno de estos nuevos retos obedecerá a la ley de las consecuencias imprevistas y tendrá efectos (para hombres de negocios, ingenieros de software y usuarios finales) que hoy no pueden predecirse. Sin embargo, los ingenieros de software pueden prepararse de-

sarrollando un proceso que sea ágil y suficientemente adaptable para que acepte los cambios profundos en la tecnología y las reglas de los negocios que seguramente surgirán en la década siguiente.

1.1.3 Software heredado

Cientos de miles de programas de cómputo caen en uno de los siete dominios amplios de aplicación que se estudiaron en la subsección anterior. Algunos de ellos son software muy nuevo, disponible para ciertos individuos, industria y gobierno. Pero otros programas son más viejos, en ciertos casos *muy* viejos.

Estos programas antiguos —que es frecuente denominar *software heredado*— han sido centro de atención y preocupación continuas desde la década de 1960. Dayani-Fard y sus colegas [Day99] describen el software heredado de la manera siguiente:

> Los sistemas de software heredado [...] fueron desarrollados hace varias décadas y han sido modificados de manera continua para que satisfagan los cambios en los requerimientos de los negocios y plataformas de computación. La proliferación de tales sistemas es causa de dolores de cabeza para las organizaciones grandes, a las que resulta costoso mantenerlos y riesgoso hacerlos evolucionar.

Liu y sus colegas [Liu98] amplían esta descripción al hacer notar que "muchos sistemas heredados continúan siendo un apoyo para las funciones básicas del negocio y son 'indispensables' para éste". Además, el software heredado se caracteriza por su longevidad e importancia crítica para el negocio.

? ¿Qué hago si encuentro un sistema heredado de mala calidad?

Desafortunadamente, en ocasiones hay otra característica presente en el software heredado: *mala calidad*.[5] Hay veces en las que los sistemas heredados tienen diseños que no son susceptibles de extenderse, código confuso, documentación mala o inexistente, casos y resultados de pruebas que nunca se archivaron, una historia de los cambios mal administrada... la lista es muy larga. A pesar de esto, dichos sistemas dan apoyo a las "funciones básicas del negocio y son indispensables para éste". ¿Qué hacer?

La única respuesta razonable es: *hacer nada*, al menos hasta que el sistema heredado tenga un cambio significativo. Si el software heredado satisface las necesidades de sus usuarios y corre de manera confiable, entonces no falla ni necesita repararse. Sin embargo, conforme pase el tiempo será frecuente que los sistemas de software evolucionen por una o varias de las siguientes razones:

? ¿Qué tipos de cambios se hacen a los sistemas heredados?

- El software debe adaptarse para que cumpla las necesidades de los nuevos ambientes del cómputo y de la tecnología.
- El software debe ser mejorado para implementar nuevos requerimientos del negocio.
- El software debe ampliarse para que sea operable con otros sistemas o bases de datos modernos.
- La arquitectura del software debe rediseñarse para hacerla viable dentro de un ambiente de redes.

Todo ingeniero de software debe reconocer que el cambio es natural. No trate de evitarlo.

Cuando ocurren estos modos de evolución, debe hacerse la reingeniería del sistema heredado (capítulo 29) para que sea viable en el futuro. La meta de la ingeniería de software moderna es "desarrollar metodologías que se basen en el concepto de evolución; es decir, el concepto de que los sistemas de software cambian continuamente, que los nuevos sistemas de software se

5 En este caso, la calidad se juzga con base en el pensamiento moderno de la ingeniería de software, criterio algo injusto, ya que algunos conceptos y principios de la ingeniería de software moderna tal vez no hayan sido bien entendidos en la época en que se desarrolló el software heredado.

desarrollan a partir de los antiguos y [...] que todo debe operar entre sí y cooperar con cada uno de los demás" [Day99].

1.2 LA NATURALEZA ÚNICA DE LAS *WEBAPPS*

Cita:

"Cuando veamos cualquier tipo de estabilización, la web se habrá convertido en algo completamente diferente."

Louis Monier

En los primeros días de la Red Mundial (entre 1990 y 1995), los *sitios web* consistían en poco más que un conjunto de archivos de hipertexto vinculados que presentaban la información con el empleo de texto y gráficas limitadas. Al pasar el tiempo, el aumento de HTML por medio de herramientas de desarrollo (XML, Java) permitió a los ingenieros de la web brindar capacidad de cómputo junto con contenido de información. Habían nacido los *sistemas y aplicaciones basados en la web*[6] (denominó a éstas en forma colectiva como *webapps*). En la actualidad, las *webapps* se han convertido en herramientas sofisticadas de cómputo que no sólo proporcionan funciones aisladas al usuario final, sino que también se han integrado con bases de datos corporativas y aplicaciones de negocios.

Como se dijo en la sección 1.1.2, las *webapps* son una de varias categorías distintas de software. No obstante, podría argumentarse que las *webapps* son diferentes. Powell [Pow98] sugiere que los sistemas y aplicaciones basados en web "involucran una mezcla entre las publicaciones impresas y el desarrollo de software, entre la mercadotecnia y la computación, entre las comunicaciones internas y las relaciones exteriores, y entre el arte y la tecnología". La gran mayoría de *webapps* presenta los siguientes atributos:

¿Qué característica diferencia las *webapps* de otro software?

Uso intensivo de redes. Una *webapp* reside en una red y debe atender las necesidades de una comunidad diversa de clientes. La red permite acceso y comunicación mundiales (por ejemplo, internet) o tiene acceso y comunicación limitados (por ejemplo, una intranet corporativa).

Concurrencia. A la *webapp* puede acceder un gran número de usuarios a la vez. En muchos casos, los patrones de uso entre los usuarios finales varían mucho.

Carga impredecible. El número de usuarios de la *webapp* cambia en varios órdenes de magnitud de un día a otro. El lunes tal vez la utilicen cien personas, el jueves quizá 10 000 usen el sistema.

Rendimiento. Si un usuario de la *webapp* debe esperar demasiado (para entrar, para el procesamiento por parte del servidor, para el formado y despliegue del lado del cliente), él o ella quizá decidan irse a otra parte.

Disponibilidad. Aunque no es razonable esperar una disponibilidad de 100%, es frecuente que los usuarios de *webapps* populares demanden acceso las 24 horas de los 365 días del año. Los usuarios en Australia o Asia quizá demanden acceso en horas en las que las aplicaciones internas de software tradicionales en Norteamérica no estén en línea por razones de mantenimiento.

Orientadas a los datos. La función principal de muchas *webapp* es el uso de hipermedios para presentar al usuario final contenido en forma de texto, gráficas, audio y video. Además, las *webapps* se utilizan en forma común para acceder a información que existe en bases de datos que no son parte integral del ambiente basado en web (por ejemplo, comercio electrónico o aplicaciones financieras).

---

6 En el contexto de este libro, el término *aplicación web* (*webapp*) agrupa todo, desde una simple página web que ayude al consumidor a calcular el pago del arrendamiento de un automóvil hasta un sitio web integral que proporcione servicios completos de viaje para gente de negocios y vacacionistas. En esta categoría se incluyen sitios web completos, funcionalidad especializada dentro de sitios web y aplicaciones de procesamiento de información que residen en internet o en una intranet o extranet.

Contenido sensible. La calidad y naturaleza estética del contenido constituye un rasgo importante de la calidad de una *webapp*.

Evolución continua. A diferencia del software de aplicación convencional que evoluciona a lo largo de una serie de etapas planeadas y separadas cronológicamente, las aplicaciones web evolucionan en forma continua. No es raro que ciertas *webapp* (específicamente su contenido) se actualicen minuto a minuto o que su contenido se calcule en cada solicitud.

Inmediatez. Aunque la *inmediatez* —necesidad apremiante de que el software llegue con rapidez al mercado— es una característica en muchos dominios de aplicación, es frecuente que las *webapps* tengan plazos de algunos días o semanas para llegar al mercado.[7]

Seguridad. Debido a que las *webapps* se encuentran disponibles con el acceso a una red, es difícil o imposible limitar la población de usuarios finales que pueden acceder a la aplicación. Con el fin de proteger el contenido sensible y brindar modos seguros de transmisión de los datos, deben implementarse medidas estrictas de seguridad a través de la infraestructura de apoyo de una *webapp* y dentro de la aplicación misma.

Estética. Parte innegable del atractivo de una *webapp* es su apariencia y percepción. Cuando se ha diseñado una aplicación para comercializar o vender productos o ideas, la estética tiene tanto que ver con el éxito como el diseño técnico.

Podría argumentarse que otras categorías de aplicaciones estudiadas en la sección 1.1.2 muestran algunos de los atributos mencionados. Sin embargo, las *webapps* casi siempre poseen todos ellos.

1.3 INGENIERÍA DE SOFTWARE

Con objeto de elaborar software listo para enfrentar los retos del siglo XXI, el lector debe aceptar algunas realidades sencillas:

PUNTO CLAVE
Entender el problema antes de dar una solución.

- El software se ha incrustado profundamente en casi todos los aspectos de nuestras vidas y, como consecuencia, el número de personas que tienen interés en las características y funciones que brinda una aplicación específica[8] ha crecido en forma notable. Cuando ha de construirse una aplicación nueva o sistema incrustado, deben escucharse muchas opiniones. Y en ocasiones parece que cada una de ellas tiene una idea un poco distinta de cuáles características y funciones debiera tener el software. *Se concluye que debe hacerse un esfuerzo concertado para entender el problema antes de desarrollar una aplicación de software.*

PUNTO CLAVE
El diseño es una actividad crucial de la ingeniería de software.

- Los requerimientos de la tecnología de la información que demandan los individuos, negocios y gobiernos se hacen más complejos con cada año que pasa. En la actualidad, grandes equipos de personas crean programas de cómputo que antes eran elaborados por un solo individuo. El software sofisticado, que alguna vez se implementó en un ambiente de cómputo predecible y autocontenido, hoy en día se halla incrustado en el interior de todo, desde la electrónica de consumo hasta dispositivos médicos o sistemas de armamento. La complejidad de estos nuevos sistemas y productos basados en computadora demanda atención cuidadosa a las interacciones de todos los elementos del sistema. *Se concluye que el diseño se ha vuelto una actividad crucial.*

7 Con las herramientas modernas es posible producir páginas web sofisticadas en unas cuantas horas.

8 En una parte posterior de este libro, llamaré a estas personas "participantes".

PUNTO CLAVE

Tanto la calidad como la facilidad de recibir mantenimiento son resultado de un buen diseño.

- Los individuos, negocios y gobiernos dependen cada vez más del software para tomar decisiones estratégicas y tácticas, así como para sus operaciones y control cotidianos. Si el software falla, las personas y empresas grandes pueden experimentar desde un inconveniente menor hasta fallas catastróficas. *Se concluye que el software debe tener alta calidad.*
- A medida que aumenta el valor percibido de una aplicación específica se incrementa la probabilidad de que su base de usuarios y longevidad también crezcan. Conforme se extienda su base de usuarios y el tiempo de uso, las demandas para adaptarla y mejorarla también crecerán. *Se concluye que el software debe tener facilidad para recibir mantenimiento.*

Estas realidades simples llevan a una conclusión: *debe hacerse ingeniería con el software en todas sus formas y a través de todos sus dominios de aplicación.* Y esto conduce al tema de este libro: *la ingeniería de software*.

Cita:

"Más que una disciplina o cuerpo de conocimientos, ingeniería es un verbo, una palabra de acción, una forma de abordar un problema."

Scott Whitmir

Aunque cientos de autores han desarrollado definiciones personales de la ingeniería de software, la propuesta por Fritz Bauer [Nau69] en la conferencia fundamental sobre el tema todavía sirve como base para el análisis:

> [La ingeniería de software es] el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales.

El lector se sentirá tentado de ampliar esta definición.[9] Dice poco sobre los aspectos técnicos de la calidad del software; no habla directamente de la necesidad de satisfacer a los consumidores ni de entregar el producto a tiempo; omite mencionar la importancia de la medición y la metrología; no establece la importancia de un proceso eficaz. No obstante, la definición de Bauer proporciona una base. ¿Cuáles son los "principios fundamentales de la ingeniería" que pueden aplicarse al desarrollo del software de computadora? ¿Cómo se desarrolla software "en forma económica" y que sea "confiable"? ¿Qué se requiere para crear programas de cómputo que trabajen con "eficiencia", no en una sino en muchas "máquinas reales" diferentes? Éstas son las preguntas que siguen siendo un reto para los ingenieros de software.

? ¿Cómo se define la ingeniería de software?

El IEEE [IEEE93a] ha desarrollado una definición más completa, como sigue:

> La ingeniería de software es: 1) La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software. 2) El estudio de enfoques según el punto 1.

Aun así, el enfoque "sistemático, disciplinado y cuantificable" aplicado por un equipo de software podría ser algo burdo para otro. Se necesita disciplina, pero también adaptabilidad y agilidad.

PUNTO CLAVE

La ingeniería de software incluye un proceso, métodos y herramientas para administrar y hacer ingeniería con el software.

La ingeniería de software es una tecnología con varias capas. Como se aprecia en la figura 1.3, cualquier enfoque de ingeniería (incluso la de software) debe basarse en un compromiso organizacional con la calidad. La administración total de la calidad, Six Sigma y otras filosofías similares[10] alimentan la cultura de mejora continua, y es esta cultura la que lleva en última instancia al desarrollo de enfoques cada vez más eficaces de la ingeniería de software. El fundamento en el que se apoya la ingeniería de software es el compromiso con la calidad.

El fundamento para la ingeniería de software es la capa *proceso*. El proceso de ingeniería de software es el aglutinante que une las capas de la tecnología y permite el desarrollo racional y

9 Consulte muchas otras definiciones en www.answers.com/topic/software-engineering#wp-_note-13.

10 En el capítulo 14 y toda la parte 3 del libro se estudia la administración de la calidad y los enfoques relacionados con ésta.

FIGURA 1.3
Capas de la ingeniería de software

WebRef
CrossTalk es un periódico que da información práctica sobre procesos, métodos y herramientas. Se encuentra en **www.stsc.hill.af.mil**

oportuno del software de cómputo. El proceso define una estructura que debe establecerse para la obtención eficaz de tecnología de ingeniería de software. El proceso de software forma la base para el control de la administración de proyectos de software, y establece el contexto en el que se aplican métodos técnicos, se generan productos del trabajo (modelos, documentos, datos, reportes, formatos, etc.), se establecen puntos de referencia, se asegura la calidad y se administra el cambio de manera apropiada.

Los *métodos* de la ingeniería de software proporcionan la experiencia técnica para elaborar software. Incluyen un conjunto amplio de tareas, como comunicación, análisis de los requerimientos, modelación del diseño, construcción del programa, pruebas y apoyo. Los métodos de la ingeniería de software se basan en un conjunto de principios fundamentales que gobiernan cada área de la tecnología e incluyen actividades de modelación y otras técnicas descriptivas.

Las *herramientas* de la ingeniería de software proporcionan un apoyo automatizado o semiautomatizado para el proceso y los métodos. Cuando se integran las herramientas de modo que la información creada por una pueda ser utilizada por otra, queda establecido un sistema llamado *ingeniería de software asistido por computadora* que apoya el desarrollo de software.

1.4 EL PROCESO DEL SOFTWARE

? ¿Cuáles son los elementos de un proceso de software?

Cita:
"Un proceso define quién hace *qué*, *cuándo* y *cómo*, para alcanzar cierto objetivo."
Ivar Jacobson, Grady Booch y James Rumbaugh

Un *proceso* es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo. Una *actividad* busca lograr un objetivo amplio (por ejemplo, comunicación con los participantes) y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usará la ingeniería de software. Una *acción* (diseño de la arquitectura) es un conjunto de tareas que producen un producto importante del trabajo (por ejemplo, un modelo del diseño de la arquitectura). Una *tarea* se centra en un objetivo pequeño pero bien definido (por ejemplo, realizar una prueba unitaria) que produce un resultado tangible.

En el contexto de la ingeniería de software, un proceso *no* es una prescripción rígida de cómo elaborar software de cómputo. Por el contrario, es un enfoque adaptable que permite que las personas que hacen el trabajo (el equipo de software) busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo. Se busca siempre entregar el software en forma oportuna y con calidad suficiente para satisfacer a quienes patrocinaron su creación y a aquellos que lo usarán.

La *estructura del proceso* establece el fundamento para el proceso completo de la ingeniería de software por medio de la identificación de un número pequeño de *actividades estructurales* que sean aplicables a todos los proyectos de software, sin importar su tamaño o complejidad. Además, la estructura del proceso incluye un conjunto de *actividades sombrilla* que son aplicables a través de todo el proceso del software. Una estructura de proceso general para la ingeniería de software consta de cinco actividades:

¿Cuáles son las cinco actividades estructurales del proceso?

Comunicación. Antes de que comience cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con el cliente (y con otros participantes).[11] Se busca entender los objetivos de los participantes respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software.

Planeación. Cualquier viaje complicado se simplifica si existe un mapa. Un proyecto de software es un viaje difícil, y la actividad de planeación crea un "mapa" que guía al equipo mientras viaja. El mapa —llamado *plan del proyecto de software*— define el trabajo de ingeniería de software al describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.

Cita:

"Einstein afirmaba que debía haber una explicación sencilla de la naturaleza porque Dios no es caprichoso o arbitrario. Al ingeniero de software no lo conforta una fe parecida. Gran parte de la complejidad que debe doblegar es de origen arbitrario."

Fred Brooks

Modelado. Ya sea usted diseñador de paisaje, constructor de puentes, ingeniero aeronáutico, carpintero o arquitecto, a diario trabaja con modelos. Crea un "bosquejo" del objeto por hacer a fin de entender el panorama general —cómo se verá arquitectónicamente, cómo ajustan entre sí las partes constituyentes y muchas características más—. Si se requiere, refina el bosquejo con más y más detalles en un esfuerzo por comprender mejor el problema y cómo resolverlo. Un ingeniero de software hace lo mismo al crear modelos a fin de entender mejor los requerimientos del software y el diseño que los satisfará.

Construcción. Esta actividad combina la generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en éste.

Despliegue. El software (como entidad completa o como un incremento parcialmente terminado) se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación.

Estas cinco actividades estructurales genéricas se usan durante el desarrollo de programas pequeños y sencillos, en la creación de aplicaciones web grandes y en la ingeniería de sistemas enormes y complejos basados en computadoras. Los detalles del proceso de software serán distintos en cada caso, pero las actividades estructurales son las mismas.

Para muchos proyectos de software, las actividades estructurales se aplican en forma iterativa a medida que avanza el proyecto. Es decir, la **comunicación,** la **planeación,** el **modelado,** la **construcción** y el **despliegue** se ejecutan a través de cierto número de repeticiones del proyecto. Cada iteración produce un *incremento del software* que da a los participantes un subconjunto de características y funcionalidad generales del software. Conforme se produce cada incremento, el software se hace más y más completo.

Las actividades estructurales del proceso de ingeniería de software son complementadas por cierto número de *actividades sombrilla*. En general, las actividades sombrilla se aplican a lo largo de un proyecto de software y ayudan al equipo que lo lleva a cabo a administrar y controlar el avance, la calidad, el cambio y el riesgo. Es común que las actividades sombrilla sean las siguientes:

PUNTO CLAVE

Las actividades sombrilla ocurren a lo largo del proceso de software y se centran sobre todo en la administración, el seguimiento y el control del proyecto.

Seguimiento y control del proyecto de software: permite que el equipo de software evalúe el progreso comparándolo con el plan del proyecto y tome cualquier acción necesaria para apegarse a la programación de actividades.

Administración del riesgo: evalúa los riesgos que puedan afectar el resultado del proyecto o la calidad del producto.

---

11 Un *participante* es cualquier persona que tenga algo que ver en el resultado exitoso del proyecto —gerentes del negocio, usuarios finales, ingenieros de software, personal de apoyo, etc.—. Rob Thomset dice en broma que "un participante es una persona que blande una estaca grande y aguda [...] Si no vez más lejos que los participantes, ya sabes dónde terminará la estaca". (N. del T.: Esta nota es un juego de palabras: *stake* significa estaca y también *parte*, y *stakeholder* es el que blande una estaca, pero también un *participante*.)

Aseguramiento de la calidad del software: define y ejecuta las actividades requeridas para garantizar la calidad del software.

Revisiones técnicas: evalúa los productos del trabajo de la ingeniería de software a fin de descubrir y eliminar errores antes de que se propaguen a la siguiente actividad.

Medición: define y reúne mediciones del proceso, proyecto y producto para ayudar al equipo a entregar el software que satisfaga las necesidades de los participantes; puede usarse junto con todas las demás actividades estructurales y sombrilla.

Administración de la configuración del software: administra los efectos del cambio a lo largo del proceso del software.

Administración de la reutilización: define criterios para volver a usar el producto del trabajo (incluso los componentes del software) y establece mecanismos para obtener componentes reutilizables.

Preparación y producción del producto del trabajo: agrupa las actividades requeridas para crear productos del trabajo, tales como modelos, documentos, registros, formatos y listas.

Cada una de estas actividades sombrilla se analiza en detalle más adelante.

PUNTO CLAVE

La adaptación del proceso de software es esencial para el éxito del proyecto.

Ya se dijo en esta sección que el proceso de ingeniería de software no es una prescripción rígida que deba seguir en forma dogmática el equipo que lo crea. Al contrario, debe ser ágil y adaptable (al problema, al proyecto, al equipo y a la cultura organizacional). Por tanto, un proceso adoptado para un proyecto puede ser significativamente distinto de otro adoptado para otro proyecto. Entre las diferencias se encuentran las siguientes:

¿Qué diferencias existen entre los modelos del proceso?

- Flujo general de las actividades, acciones y tareas, así como de las interdependencias entre ellas
- Grado en el que las acciones y tareas están definidas dentro de cada actividad estructural
- Grado en el que se identifican y requieren los productos del trabajo
- Forma en la que se aplican las actividades de aseguramiento de la calidad
- Manera en la que se realizan las actividades de seguimiento y control del proyecto
- Grado general de detalle y rigor con el que se describe el proceso
- Grado con el que el cliente y otros participantes se involucran con el proyecto
- Nivel de autonomía que se da al equipo de software
- Grado con el que son prescritos la organización y los roles del equipo

Cita:

"Siento que una receta es sólo un tema que una cocinera inteligente ejecuta con una variación en cada ocasión."

Madame Benoit

En la parte 1 de este libro, se examinará el proceso de software con mucho detalle. Los *modelos de proceso prescriptivo* (capítulo 2) enfatizan la definición, la identificación y la aplicación detalladas de las actividades y tareas del proceso. Su objetivo es mejorar la calidad del sistema, desarrollar proyectos más manejables, hacer más predecibles las fechas de entrega y los costos, y guiar a los equipos de ingenieros de software cuando realizan el trabajo que se requiere para construir un sistema. Desafortunadamente, ha habido casos en los que estos objetivos no se han logrado. Si los modelos prescriptivos se aplican en forma dogmática y sin adaptación, pueden incrementar el nivel de burocracia asociada con el desarrollo de sistemas basados en computadora y crear inadvertidamente dificultades para todos los participantes.

¿Qué caracteriza a un proceso "ágil"?

Los *modelos de proceso ágil* (capítulo 3) ponen el énfasis en la "agilidad" del proyecto y siguen un conjunto de principios que conducen a un enfoque más informal (pero no menos efectivo, dicen sus defensores) del proceso de software. Por lo general, se dice que estos modelos del proceso son "ágiles" porque acentúan la maniobrabilidad y la adaptabilidad. Son apropiados para muchos tipos de proyectos y son útiles en particular cuando se hace ingeniería sobre aplicaciones web.

1.5 La práctica de la ingeniería de software

WebRef

En la dirección **www.literateprogramming.com** se encuentran varias citas provocativas sobre la práctica de la ingeniería de software.

En la sección 1.4 se introdujo un modelo general de proceso de software compuesto de un conjunto de actividades que establecen una estructura para la práctica de la ingeniería de software. Las actividades estructurales generales —**comunicación, planeación, modelado, construcción** y **despliegue**— y las actividades sombrilla establecen el esqueleto de la arquitectura para el trabajo de ingeniería de software. Pero, ¿cómo entra aquí la práctica de la ingeniería de software? En las secciones que siguen, el lector obtendrá la comprensión básica de los conceptos y principios generales que se aplican a las actividades estructurales.[12]

1.5.1 La esencia de la práctica

CONSEJO

Podría decirse que el enfoque de Polya es simple sentido común. Es verdad. Pero es sorprendente la frecuencia con la que el sentido común es poco común en el mundo del software.

En un libro clásico, *How to Solve It*, escrito antes de que existieran las computadoras modernas, George Polya [Pol45] describió la esencia de la solución de problemas y, en consecuencia, la esencia de la práctica de la ingeniería de software:

1. *Entender el problema* (comunicación y análisis).
2. *Planear la solución* (modelado y diseño del software).
3. *Ejecutar el plan* (generación del código).
4. *Examinar la exactitud del resultado* (probar y asegurar la calidad).

En el contexto de la ingeniería de software, estas etapas de sentido común conducen a una serie de preguntas esenciales [adaptado de Pol45]:

Entender el problema. En ocasiones es difícil de admitir, pero la mayor parte de nosotros adoptamos una actitud de orgullo desmedido cuando se nos presenta un problema. Escuchamos por unos segundos y después pensamos: *Claro, sí, entiendo, resolvamos esto.* Desafortunadamente, entender no siempre es fácil. Es conveniente dedicar un poco de tiempo a responder algunas preguntas sencillas:

- *¿Quiénes tienen que ver con la solución del problema?* Es decir, ¿quiénes son los participantes?
- *¿Cuáles son las incógnitas?* ¿Cuáles datos, funciones y características se requieren para resolver el problema en forma apropiada?
- *¿Puede fraccionarse el problema?* ¿Es posible representarlo con problemas más pequeños que sean más fáciles de entender?
- *¿Es posible representar gráficamente el problema?* ¿Puede crearse un modelo de análisis?

Cita:

"En la solución de cualquier problema hay un grano de descubrimiento."

George Polya

Planear la solución. Ahora entiende el problema (o es lo que piensa) y no puede esperar para escribir el código. Antes de hacerlo, cálmese un poco y haga un pequeño diseño:

- *¿Ha visto antes problemas similares?* ¿Hay patrones reconocibles en una solución potencial? ¿Hay algún software existente que implemente los datos, funciones y características que se requieren?
- *¿Ha resuelto un problema similar?* Si es así, ¿son reutilizables los elementos de la solución?
- *¿Pueden definirse problemas más pequeños?* Si así fuera, ¿hay soluciones evidentes para éstos?

12 El lector debería volver a consultar las secciones de este capítulo a medida que en el libro se describan en específico los métodos y las actividades sombrilla de la ingeniería de software.

- *¿Es capaz de representar una solución en una forma que lleve a su implementación eficaz?* ¿Es posible crear un modelo del diseño?

Ejecutar el plan. El diseño que creó sirve como un mapa de carreteras para el sistema que quiere construir. Puede haber desviaciones inesperadas y es posible que descubra un camino mejor a medida que avanza, pero el "plan" le permitirá proceder sin que se pierda.

- *¿Se ajusta la solución al plan?* ¿El código fuente puede apegarse al modelo del diseño?
- *¿Es probable que cada parte componente de la solución sea correcta?* ¿El diseño y código se han revisado o, mejor aún, se han hecho pruebas respecto de la corrección del algoritmo?

Examinar el resultado. No se puede estar seguro de que la solución sea perfecta, pero sí de que se ha diseñado un número suficiente de pruebas para descubrir tantos errores como sea posible.

- *¿Puede probarse cada parte componente de la solución?* ¿Se ha implementado una estrategia razonable para hacer pruebas?
- *¿La solución produce resultados que se apegan a los datos, funciones y características que se requieren?* ¿El software se ha validado contra todos los requerimientos de los participantes?

No debiera sorprender que gran parte de este enfoque tenga que ver con el sentido común. En realidad, es razonable afirmar que un enfoque de sentido común para la ingeniería de software hará que nunca se extravíe.

1.5.2 Principios generales

El diccionario define la palabra *principio* como "una ley importante o suposición que subyace y se requiere en un sistema de pensamiento". En este libro se analizarán principios en muchos niveles distintos de abstracción. Algunos se centran en la ingeniería de software como un todo, otros consideran una actividad estructural general específica (por ejemplo, **comunicación**), y otros más se centran en acciones de la ingeniería de software (por ejemplo, diseño de la arquitectura) o en tareas técnicas (escribir un escenario para el uso). Sin importar su nivel de enfoque, los principios lo ayudarán a establecer un conjunto de herramientas mentales para una práctica sólida de la ingeniería de software. Ésa es la razón de que sean importantes.

David Hooker [Hoo96] propuso siete principios que se centran en la práctica de la ingeniería de software como un todo. Se reproducen en los párrafos siguientes:[13]

Antes de comenzar un proyecto de software, asegúrese de que el software tenga un propósito para el negocio y que los usuarios perciben valor en él.

Primer principio: *La razón de que exista todo*

Un sistema de software existe por una razón: *dar valor a sus usuarios.* Todas las decisiones deben tomarse teniendo esto en mente. Antes de especificar un requerimiento del sistema, antes de notar la funcionalidad de una parte de él, antes de determinar las plataformas del hardware o desarrollar procesos, plantéese preguntas tales como: "¿Esto agrega valor real al sistema?" Si la respuesta es "no", entonces no lo haga. Todos los demás principios apoyan a éste.

Segundo principio: *MSE (Mantenlo sencillo, estúpido...)*

El diseño de software no es un proceso caprichoso. Hay muchos factores por considerar en cualquier actividad de diseño. *Todo diseño debe ser tan simple como sea posible, pero no más*.

13 Reproducido con permiso del autor [Hoo96]. Hooker define algunos patrones para estos principios en http://c2.com/cgi/wiki?SevenPrinciplesOfSoftwareDevelopment.

Cita:

"Hay cierta majestad en la sencillez, que es con mucho todo lo que adorna al ingenio."

Papa Alejandro (1688-1744)

Esto facilita conseguir un sistema que sea comprendido más fácilmente y que sea susceptible de recibir mantenimiento, lo que no quiere decir que en nombre de la simplicidad deban descartarse características o hasta rasgos internos. En realidad, los diseños más elegantes por lo general son los más simples. Simple tampoco significa "rápido y sucio". La verdad es que con frecuencia se requiere mucha reflexión y trabajo con iteraciones múltiples para poder simplificar. La recompensa es un software más fácil de mantener y menos propenso al error.

Tercer principio: *Mantener la visión*

Una *visión clara es esencial para el éxito de un proyecto de software*. Sin ella, casi infaliblemente el proyecto terminará siendo un ser "con dos [o más mentes]". Sin integridad conceptual, un sistema está amenazado de convertirse en una urdimbre de diseños incompatibles unidos por tornillos del tipo equivocado [...] Comprometer la visión de la arquitectura de un sistema de software debilita y, finalmente hará que colapsen incluso los sistemas bien diseñados. Tener un arquitecto que pueda para mantener la visión y que obligue a su cumplimiento garantiza un proyecto de software muy exitoso.

Cuarto principio: *Otros consumirán lo que usted produce*

¡PUNTO CLAVE

Si el software tiene valor, cambiará durante su vida útil. Por esa razón, debe construirse de forma que sea fácil darle mantenimiento.

Rara vez se construye en el vacío un sistema de software con fortaleza industrial. En un modo u otro, alguien más lo usará, mantendrá, documentará o, de alguna forma, dependerá de su capacidad para entender el sistema. Así que *siempre establezca especificaciones, diseñe e implemente con la seguridad de que alguien más tendrá que entender lo que usted haga*. La audiencia para cualquier producto de desarrollo de software es potencialmente grande. Elabore especificaciones con la mirada puesta en los usuarios. Diseñe con los implementadores en mente. Codifique pensando en aquellos que deben dar mantenimiento y ampliar el sistema. Alguien debe depurar el código que usted escriba, y eso lo hace usuario de su código. Hacer su trabajo más fácil agrega valor al sistema.

Quinto principio: *Ábrase al futuro*

Un sistema con larga vida útil tiene más valor. En los ambientes de cómputo actuales, donde las especificaciones cambian de un momento a otro y las plataformas de hardware quedan obsoletas con sólo unos meses de edad, es común que la vida útil del software se mida en meses y no en años. Sin embargo, los sistemas de software con verdadera "fortaleza industrial" deben durar mucho más tiempo. Para tener éxito en esto, los sistemas deben ser fáciles de adaptar a ésos y otros cambios. Los sistemas que lo logran son los que se diseñaron para ello desde el principio. *Nunca diseñe sobre algo iniciado.* Siempre pregunte: "¿qué pasa si...?" y prepárese para todas las respuestas posibles mediante la creación de sistemas que resuelvan el problema general, no sólo uno específico.[14] Es muy posible que esto lleve a volver a usar un sistema completo.

Sexto principio: *Planee por anticipado la reutilización*

La reutilización ahorra tiempo y esfuerzo.[15] Al desarrollar un sistema de software, lograr un alto nivel de reutilización es quizá la meta más difícil de lograr. La reutilización del código y de los diseños se ha reconocido como uno de los mayores beneficios de usar tecnologías orientadas a objetos. Sin embargo, la recuperación de esta inversión no es automática. Para reforzar las posibilidades de la reutilización que da la programación orientada a objetos [o la

14 Es peligroso llevar este consejo a los extremos. Diseñar para resolver "el problema general" en ocasiones requiere compromisos de rendimiento y puede volver ineficientes las soluciones específicas.

15 Aunque esto es verdad para aquellos que reutilizan software en proyectos futuros, volver a usar puede ser caro para quienes deben diseñar y elaborar componentes reutilizables. Los estudios indican que diseñar y construir componentes reutilizables llega a costar entre 25 y 200% más que el software buscado. En ciertos casos no se justifica la diferencia de costos.

convencional], se requiere reflexión y planeación. Hay muchas técnicas para incluir la reutilización en cada nivel del proceso de desarrollo del sistema… *La planeación anticipada en busca de la reutilización disminuye el costo e incrementa el valor tanto de los componentes reutilizables como de los sistemas en los que se incorpora.*

Séptimo principio: *¡Piense!*

Este último principio es tal vez el que más se pasa por alto. *Pensar en todo con claridad antes de emprender la acción casi siempre produce mejores resultados*. Cuando se piensa en algo es más probable que se haga bien. Asimismo, también se gana conocimiento al pensar cómo volver a hacerlo bien. Si usted piensa en algo y aun así lo hace mal, eso se convierte en una experiencia valiosa. Un efecto colateral de pensar es aprender a reconocer cuando no se sabe algo, punto en el que se puede investigar la respuesta. Cuando en un sistema se han puesto pensamientos claros, el valor se manifiesta. La aplicación de los primeros seis principios requiere pensar con intensidad, por lo que las recompensas potenciales son enormes.

Si todo ingeniero y equipo de software tan sólo siguiera los siete principios de Hooker, se eliminarían muchas de las dificultades que se experimentan al construir sistemas complejos basados en computadora.

1.6 MITOS DEL SOFTWARE

Cita:

"En ausencia de estándares significativos, una industria nueva como la del software depende sólo del folklore."

Tom DeMarco

Los mitos del software —creencias erróneas sobre éste y sobre el proceso que se utiliza para obtenerlo— se remontan a los primeros días de la computación. Los mitos tienen cierto número de atributos que los hacen insidiosos. Por ejemplo, parecen enunciados razonables de hechos (a veces contienen elementos de verdad), tienen una sensación intuitiva y es frecuente que los manifiesten profesionales experimentados que "conocen la historia".

En la actualidad, la mayoría de profesionales de la ingeniería de software reconocen los mitos como lo que son: actitudes equivocadas que han ocasionado serios problemas a los administradores y a los trabajadores por igual. Sin embargo, las actitudes y hábitos antiguos son difíciles de modificar, y persisten algunos remanentes de los mitos del software.

WebRef

La Software Project Managers Network (Red de Gerentes de Proyectos de Software), en **www.spmn.com**, lo ayuda a eliminar éstos y otros mitos.

Mitos de la administración. Los gerentes que tienen responsabilidades en el software, como los de otras disciplinas, con frecuencia se hallan bajo presión para cumplir el presupuesto, mantener la programación de actividades sin desvíos y mejorar la calidad. Así como la persona que se ahoga se agarra de un clavo ardiente, no es raro que un gerente de software sostenga la creencia en un mito del software si eso disminuye la presión a que está sujeto (incluso de manera temporal).

Mito: *Tenemos un libro lleno de estándares y procedimientos para elaborar software. ¿No le dará a mi personal todo lo que necesita saber?*

Realidad: Tal vez exista el libro de estándares, pero ¿se utiliza? ¿Saben de su existencia los trabajadores del software? ¿Refleja la práctica moderna de la ingeniería de software? ¿Es completo? ¿Es adaptable? ¿Está dirigido a mejorar la entrega a tiempo y también se centra en la calidad? En muchos casos, la respuesta a todas estas preguntas es "no".

Mito: *Si nos atrasamos, podemos agregar más programadores y ponernos al corriente (en ocasiones, a esto se le llama "concepto de la horda de mongoles").*

Realidad: El desarrollo del software no es un proceso mecánico similar a la manufactura. En palabras de Brooks [Bro95]: "agregar personal a un proyecto de software atrasado lo atrasará más". Al principio, esta afirmación parece ir contra la intuición. Sin embargo, a medida que se agregan personas, las que ya se

encontraban trabajando deben dedicar tiempo para enseñar a los recién llegados, lo que disminuye la cantidad de tiempo dedicada al esfuerzo de desarrollo productivo. Pueden agregarse individuos, pero sólo en forma planeada y bien coordinada.

Mito: *Si decido subcontratar el proyecto de software a un tercero, puedo descansar y dejar que esa compañía lo elabore.*

Realidad: Si una organización no comprende cómo administrar y controlar proyectos de software internamente, de manera invariable tendrá dificultades cuando subcontrate proyectos de software.

Mitos del cliente. El cliente que requiere software de computadora puede ser la persona en el escritorio de al lado, un grupo técnico en el piso inferior, el departamento de mercadotecnia y ventas, o una compañía externa que solicita software mediante un contrato. En muchos casos, el cliente sostiene mitos sobre el software porque los gerentes y profesionales de éste hacen poco para corregir la mala información. Los mitos generan falsas expectativas (por parte del cliente) y, en última instancia, la insatisfacción con el desarrollador.

Trabaje muy duro para entender qué es lo que tiene que hacer antes de empezar. Quizás no pueda desarrollarlo a detalle, pero entre más sepa, menor será el riesgo que tome.

Mito: *Para comenzar a escribir programas, es suficiente el enunciado general de los objetivos —podremos entrar en detalles más adelante.*

Realidad: Aunque no siempre es posible tener el enunciado exhaustivo y estable de los requerimientos, un "planteamiento de objetivos" ambiguo es una receta para el desastre. Los requerimientos que no son ambiguos (que por lo general se obtienen en forma iterativa) se desarrollan sólo por medio de una comunicación eficaz y continua entre el cliente y el desarrollador.

Mito: *Los requerimientos del software cambian continuamente, pero el cambio se asimila con facilidad debido a que el software es flexible.*

Realidad: Es verdad que los requerimientos del software cambian, pero el efecto que los cambios tienen varía según la época en la que se introducen. Cuando se solicitan al principio cambios en los requerimientos (antes de que haya comenzado el diseño o la elaboración de código), el efecto sobre el costo es relativamente pequeño.[16] Sin embargo, conforme pasa el tiempo, el costo aumenta con rapidez: los recursos ya se han comprometido, se ha establecido la estructura del diseño y el cambio ocasiona perturbaciones que exigen recursos adicionales y modificaciones importantes del diseño.

Siempre que piense que no hay tiempo para la ingeniería de software, pregúntese: "¿tendremos tiempo de hacerlo otra vez?".

Mitos del profesional. Los mitos que aún sostienen los trabajadores del software han sido alimentados por más de 50 años de cultura de programación. Durante los primeros días, la programación se veía como una forma del arte. Es difícil que mueran los hábitos y actitudes arraigados.

Mito: *Una vez que escribimos el programa y hacemos que funcione, nuestro trabajo ha terminado.*

Realidad: Alguien dijo alguna vez que "entre más pronto se comience a 'escribir el código', más tiempo tomará hacer que funcione". Los datos de la industria indican que entre 60 y 80% de todo el esfuerzo dedicado al software ocurrirá después de entregarlo al cliente por primera vez.

Mito: *Hasta que no se haga "correr" el programa, no hay manera de evaluar su calidad.*

16 Muchos ingenieros de software han adoptado un enfoque "ágil" que asimila los cambios en forma gradual y creciente, con lo que controlan su efecto y costo. Los métodos ágiles se estudian en el capítulo 3.

Realidad: Uno de los mecanismos más eficaces de asegurar la calidad del software puede aplicarse desde la concepción del proyecto: *la revisión técnica*. Las revisiones del software (descritas en el capítulo 15) son un "filtro de la calidad" que se ha revelado más eficaz que las pruebas para encontrar ciertas clases de defectos de software.

Mito: *El único producto del trabajo que se entrega en un proyecto exitoso es el programa que funciona.*

Realidad: Un programa que funciona sólo es una parte de una configuración de software que incluye muchos elementos. Son varios los productos terminados (modelos, documentos, planes) que proporcionan la base de la ingeniería exitosa y, lo más importante, que guían el apoyo para el software.

Mito: *La ingeniería de software hará que generemos documentación voluminosa e innecesaria, e invariablemente nos retrasará.*

Realidad: La ingeniería de software no consiste en producir documentos. Se trata de crear un producto de calidad. La mejor calidad conduce a menos repeticiones, lo que da como resultado tiempos de entrega más cortos.

Muchos profesionales del software reconocen la falacia de los mitos mencionados. Es lamentable que las actitudes y métodos habituales nutran la administración y las prácticas técnicas deficientes, aun cuando la realidad dicta un enfoque mejor. El primer paso hacia la formulación de soluciones prácticas para la ingeniería de software es el reconocimiento de las realidades en este campo.

1.7 Cómo comienza todo

Todo proyecto de software se desencadena por alguna necesidad de negocios: la de corregir un defecto en una aplicación existente, la de adaptar un "sistema heredado" a un ambiente de negocios cambiante, la de ampliar las funciones y características de una aplicación ya existente o la necesidad de crear un producto, servicio o sistema nuevo.

Al comenzar un proyecto de software, es frecuente que las necesidades del negocio se expresen de manera informal como parte de una simple conversación. La plática que se presenta en el recuadro que sigue es muy común.

CasaSegura[17]

Cómo se inicia un proyecto

La escena: Sala de juntas en CPI Corporation, empresa (ficticia) que manufactura productos de consumo para uso doméstico y comercial.

Participantes: Mal Golden, alto directivo de desarrollo de productos; Lisa Pérez, gerente comercial; Lee Warren, gerente de ingeniería; Joe Camalleri, VP ejecutivo, desarrollo de negocios.

La conversación:

Joe: Oye, Lee, ¿qué es eso que oí acerca de que tu gente va a desarrollar no sé qué? ¿Una caja inalámbrica universal general?

Lee: Es sensacional… más o menos del tamaño de una caja de cerillos pequeña… podemos conectarla a sensores de todo tipo, una cámara digital… a cualquier cosa. Usa el protocolo 802.11g inalámbrico. Permite el acceso a la salida de dispositivos sin cables. Pensamos que llevará a toda una nueva generación de productos.

Joe: ¿Estás de acuerdo, Mal?

Mal: Sí. En realidad, con las ventas tan planas que hemos tenido este año necesitamos algo nuevo. Lisa y yo hemos hecho algo de investigación del mercado y pensamos que tenemos una línea de productos que podría ser algo grande.

17 El proyecto *CasaSegura* se usará en todo el libro para ilustrar los entretelones de un equipo de proyecto que elabora un producto de software. La compañía, el proyecto y las personas son ficticias, pero las situaciones y problemas son reales.

Joe: ¿Cuán grande… tanto como el renglón de utilidades?

Mal (que evita el compromiso directo): Cuéntale nuestra idea, Lisa.

Lisa: Es toda una nueva generación que hemos llamado "productos para la administración del hogar". Le dimos el nombre de *CasaSegura*. Usan la nueva interfaz inalámbrica, proporcionan a los dueños de viviendas o pequeños negocios un sistema controlado por su PC —seguridad del hogar, vigilancia, control de aparatos y equipos—, tú sabes, apaga el aire acondicionado cuando sales de casa, esa clase de cosas.

Lee (dando un brinco): La oficina de ingeniería hizo un estudio de factibilidad técnica de esta idea, Joe. Es algo realizable con un costo bajo de manufactura. La mayor parte del hardware es de línea. Queda pendiente el software, pero no es algo que no podamos hacer.

Joe: Interesante. Pero pregunté sobre las utilidades.

Mal: Las PC han penetrado a 70 por ciento de los hogares de Estados Unidos. Si lo vendemos en el precio correcto, podría ser una aplicación sensacional. Nadie tiene nuestra caja inalámbrica… somos dueños. Nos adelantaremos dos años a la competencia. ¿Las ganancias? Quizá tanto como 30 a 40 millones de dólares en el segundo año.

Joe (sonriente): Llevemos esto al siguiente nivel. Estoy interesado.

Con excepción de una referencia casual, el software no se mencionó en la conversación. Y, sin embargo, es lo que hará triunfar o fracasar la línea de productos *CasaSegura*. El esfuerzo de ingeniería tendrá éxito sólo si también lo tiene el software de *CasaSegura*. El mercado aceptará el producto sólo si el software incrustado en éste satisface las necesidades del cliente (aún no establecidas). En muchos de los capítulos siguientes continuaremos el avance de la ingeniería del software en *CasaSegura*.

1.8 Resumen

El software es un elemento clave en la evolución de sistemas y productos basados en computadoras, y una de las tecnologías más importantes en todo el mundo. En los últimos 50 años, el software ha pasado de ser la solución de un problema especializado y herramienta de análisis de la información a una industria en sí misma. No obstante, aún hay problemas para desarrollar software de alta calidad a tiempo y dentro del presupuesto asignado.

El software —programas, datos e información descriptiva— se dirige a una gama amplia de tecnología y campos de aplicación. El software heredado sigue planteando retos especiales a quienes deben darle mantenimiento.

Los sistemas y aplicaciones basados en web han evolucionado de simples conjuntos de contenido de información a sistemas sofisticados que presentan una funcionalidad compleja y contenido en multimedios. Aunque dichas *webapps* tienen características y requerimientos únicos, son software.

La ingeniería de software incluye procesos, métodos y herramientas que permiten elaborar a tiempo y con calidad sistemas complejos basados en computadoras. El proceso de software incorpora cinco actividades estructurales: comunicación, planeación, modelado, construcción y despliegue que son aplicables a todos los proyectos de software. La práctica de la ingeniería de software es una actividad para resolver problemas, que sigue un conjunto de principios fundamentales.

Muchos mitos del software todavía hacen que administradores y trabajadores se equivoquen, aun cuando ha aumentado nuestro conocimiento colectivo del software y las tecnologías requeridas para elaborarlo. Conforme el lector aprenda más sobre ingeniería de software, comenzará a entender por qué deben rebatirse estos mitos cada vez que surjan.

Problemas y puntos por evaluar

1.1. Dé al menos cinco ejemplos de la forma en que se aplica la ley de las consecuencias imprevistas al software de cómputo.

1.2. Diga algunos ejemplos (tanto positivos como negativos) que indiquen el efecto del software en nuestra sociedad.

1.3. Desarrolle sus propias respuestas a las cinco preguntas planteadas al principio de la sección 1.1. Analícelas con sus compañeros estudiantes.

1.4. Muchas aplicaciones modernas cambian con frecuencia, antes de que se presenten al usuario final y después de que la primera versión ha entrado en uso. Sugiera algunos modos de elaborar software para detener el deterioro que produce el cambio.

1.5. Considere las siete categorías de software presentadas en la sección 1.1.2. ¿Piensa que puede aplicarse a cada una el mismo enfoque de ingeniería de software? Explique su respuesta.

1.6. La figura 1.3 muestra las tres capas de la ingeniería de software arriba de otra llamada "compromiso con la calidad". Esto implica un programa de calidad organizacional como el enfoque de la administración total de la calidad. Haga un poco de investigación y desarrolle los lineamientos de los elementos clave de un programa para la administración de la calidad.

1.7. ¿Es aplicable la ingeniería de software cuando se elaboran *webapps*? Si es así, ¿cómo puede modificarse para que asimile las características únicas de éstas?

1.8. A medida que el software gana ubicuidad, los riesgos para el público (debidos a programas defectuosos) se convierten en motivo de preocupación significativa. Desarrolle un escenario catastrófico pero realista en el que la falla de un programa de cómputo pudiera ocasionar un gran daño (económico o humano).

1.9. Describa con sus propias palabras una estructura de proceso. Cuando se dice que las actividades estructurales son aplicables a todos los proyectos, ¿significa que se realizan las mismas tareas en todos los proyectos sin que importe su tamaño y complejidad? Explique su respuesta.

1.10. Las actividades sombrilla ocurren a través de todo el proceso del software. ¿Piensa usted que son aplicables por igual a través del proceso, o que algunas se concentran en una o más actividades estructurales?

1.11. Agregue dos mitos adicionales a la lista presentada en la sección 1.6. También diga la realidad que acompaña al mito.

Lecturas adicionales y fuentes de información[18]

Hay literalmente miles de libros escritos sobre software de cómputo. La gran mayoría analiza lenguajes de programación o aplicaciones de software, pero algunos estudian al software en sí mismo. Pressman y Herron (*Software Shock*, Dorset House, 1991) presentaron un estudio temprano (dirigido a las personas comunes) sobre el software y la forma en la que lo elaboran los profesionales. El libro de Negroponte que se convirtió en un éxito de ventas (*Being Digital*, Alfred A. Knopf, Inc., 1995) describe el panorama de la computación y su efecto general en el siglo XXI. DeMarco (*Why Does Software Cost So Much?*, Dorset House, 1995) ha producido varios ensayos amenos y profundos sobre el software y el proceso con el que se elabora.

Minasi (*The Software Conspiracy: Why Software Companies Put out Faulty Products, How They Can Hurt You, and What You Can Do*, McGraw-Hill, 2000) afirma que el "flagelo moderno" de los errores en el software puede eliminarse y sugiere formas de lograrlo. Compaine (*Digital Divide: Facing A Crisis or Creating a Myth*, MIT Press, 2001) asegura que la "división" entre aquellos que tienen acceso a recursos de la información (por ejemplo, la web) y los que no lo tienen se está estrechando conforme avanzamos en la primera década de este siglo. Los libros escritos por Greenfield (*Everyware: The Dawning Age of Ubiquitous Computing*, New Riders Publishing, 2006) y Loke (*Context-Aware Pervasive Systems: Architectures for a New Breed of Applications*, Auerbach, 2006) introducen el concepto de software de "mundo abierto" y predicen un ambiente inalámbrico en el que el software deba adaptarse a los requerimientos que surjan en tiempo real.

---

18 La sección de "Lecturas adicionales y fuentes de información" que se presenta al final de cada capítulo expone un panorama breve de fuentes impresas que ayudan a aumentar la comprensión de los principales temas presentados. El autor ha creado un sitio web para apoyar al libro *Ingeniería de software: enfoque del profesional* en www.mhhe.com/compsci/pressman. Entre los muchos temas que se abordan en dicho sitio, se encuentran desde los recursos de la ingeniería de software capítulo por capítulo hasta información basada en web que complementa el material presentado. Como parte de esos recursos se halla un vínculo hacia Amazon.com para cada libro citado en esta sección.

El estado actual de la ingeniería y del proceso de software se determina mejor a partir de publicaciones tales como *IEEE Software, IEEE Computer, CrossTalk* y *IEEE Transactions on Software Engineering*. Publicaciones periódicas como *Application Development Trends* y *Cutter IT Journal* con frecuencia contienen artículos sobre temas de ingeniería de software. La disciplina se "resume" cada año en *Proceeding of the International Conference on Software Engineering*, patrocinada por IEEE y ACM, y se analiza a profundidad en revistas tales como *ACM Transactions on Software Engineering and Methodology, ACM Software Engineering Notes* y *Annals of Software Engineering*. Hay decenas de miles de sitios web dedicados a la ingeniería y al proceso de software.

En años recientes se han publicado muchos libros que abordan el proceso y la ingeniería de software. Algunos presentan un panorama de todo el proceso, mientras otros profundizan en algunos temas importantes y omiten otros. Entre los más populares (¡además del que tiene usted en sus manos!) se encuentran los siguientes:

Abran, A., and J. Moore, *SWEBOK: Guide to the Software Engineering Body of Knowledge,* IEEE, 2002.

Andersson, E., et al., *Software Engineering for Internet Applications,* The MIT Press, 2006.

Christensen, M., and R. Thayer, *A Project Manager's Guide to Software Engineering Best Practices,* IEEE-CS Press (Wiley), 2002.

Glass, R., *Fact and Fallacies of Software Engineering,* Addison-Wesley, 2002.

Jacobson, I., *Object-Oriented Software Engineering: A Use Case Driven Approach,* 2d ed., Addison-Wesley, 2008.

Jalote, P., *An Integrated Approach to Software Engineering,* Springer, 2006.

Pfleeger, S., *Software Engineering: Theory and Practice,* 3d ed., Prentice-Hall, 2005.

Schach, S., *Object-Oriented and Classical Software Engineering,* 7th ed., McGraw-Hill, 2006.

Sommerville, I., *Software Engineering,* 8th ed., Addison-Wesley, 2006.

Tsui, F., and O. Karam, *Essentials of Software Engineering,* Jones & Bartlett Publishers, 2006.

En las últimas décadas, son muchos los estándares para la ingeniería de software que han sido publicados por IEEE, ISO y sus organizaciones. Moore (*The Road Map to Software Engineering: A Standards-Based Guide*, Wiley-IEEE Computer Society Press, 2006) proporciona una revisión útil de los estándares relevantes y la forma en la que se aplican a proyectos reales.

En internet se encuentra disponible una amplia variedad de fuentes acerca de la ingeniería y el proceso de software. Una lista actualizada de referencias en la Red Mundial que son útiles para el proceso de software se encuentra en el sitio web del libro, en la dirección **www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm**.