1. Génesis de la sociología

1.1 Definición de la sociología

Vivimos hoy en un mundo que es enormemente preocupante, pero lleno de las más extraordinarias promesas para el futuro, es un mundo pletórico de cambios, sin embargo, tenemos posibilidades de controlar nuestro destino, de conformar nuestras vidas para lo mejor: ¿ cómo surgió este mundo?, ¿Por qué

son nuestras condiciones de vida tan diferentes de las de nuestros antepasados?, ¿ Qué dirección tomará el cambio en el futuro ¿ , estas son las cuestiones que tienen preocupación primordial de la sociología, una disciplina que, por consiguiente, tiene que desempeñar un papel fundamental la cultura intelectual moderna.

La sociología es la ciencia social por excelencia, pretende, desde sus padres fundadores hasta nosotros mismos nada menos que comprender la sociedad, la acción social, la conducta de los hombres y los motivos que los mueven. La ciencia buscada por los sociólogos, y en parte encontrada, desea comprender el enigma y la peculiaridad del hombre en su vida social e histórica.

La sociología es un intento de aplicar los métodos de la ciencia al estudio del hombre como ser social y a la sociedad.

Se basa en el supuesto, común a todas las ciencias, de que el método científico puede contribuir al conocimiento y dominio del hombre sobre el mundo que le rodea.

En el caso de las ciencias sociales, esto se concreta en la confianza de que es posible profundizar en el carácter social del hombre y en que el conocimiento adquirido sea útil.

Una de las ideas más importantes en la génesis de la sociología, fue el descubrimiento de que el medio generado por la interacción entre los hombres, pero que una vez que exista y con respecto al ser humano en singular, adquiere el carácter de realidad independiente y le condiciona en todas sus formas de hacer y pensar. En este sentido, la sociología pretende comprender y explicar el comportamiento humano.

El origen de la sociología está en las conmociones de la sociedad europea desde el último tercio del siglo XVIII, hubo de efectuarse una nueva organización de la sociedad bajo la dirección del espíritu científico enérgicamente desarrollado en el siglo XVIII, partiendo de esta necesidad había de establecerle la conexión del sistema entero de las verdades científicas, que se eleva desde la matemática y como su último término habla de fundarse la nueva ciencia redentora de la sociedad Condorcet y Saint Simon fueron los precursores, Comte, el fundador de esa ciencia general de la sociedad.

Si reflexionamos sobre la sociedad y la vida social muy próximas a la teoría sociológica las ha habido desde siempre, los orígenes de la teoría sociológica deben situarse en la Francia revolucionaria, es decir, a finales del siglo XVIII, debido a una serie de alteraciones de los marcos sociales que favorecieron extraordinariamente el planteamiento de la teoría de la sociedad en el sentido de teoría sociológica, a partir de entonces se tratará de crear una nueva manera de estudiar al ser humano y a la sociedad. Los problemas radicalmente nuevos planteados por la nueva sociedad que estaba surgiendo, pusieron de esto las respuestas heredadas y produjeron el nacimiento de la sociología.

La sociología como ciencia nace en la Francia revolucionaria La sociedad europeas después de la Revolución Francesa era diferente, tres hechos fundamentales permiten explicar el origen de la sociología y comprender su proyecto y estas son:

Las alternativas en el contexto económico social

Las modificaciones en el marco político

El desarrollo de la Ciencia moderna y de la tecnología.

La sociología esta considerada como un estudio de tipo sistemático de la sociedad humana así como el comportamiento humano en los contextos sociales, sistemático por que se dedica al estudio de cómo son creadas las sociedades como se mantienen y porque cambian y todo esto como afecta el comportamiento de los individuos.

La sociología se pregunta el que y porque, que hace la gente y porque lo hace que lo conduce a hacer las cosas.

Mediante la sociología se observa la interacción de los individuos ya que para desarrollar cualquier actividad es necesario la intervención de estos, los individuos nos encontramos en construcción y reconstrucción de grupos y con esto reflejamos los acontecimientos sociales.

Cada uno de nosotros nos encontramos divididos en categorías como son el sexo la edad, la raza, ocupación etc. Y con esto definimos quienes somos dentro de la sociedad en la que nos toco nacer y vivir.

Uno de los fines de la sociología es que la estructura de la sociedad afecta las actitudes y el comportamiento de la gente y desgraciadamente esto no se

percibe en forma consciente sino hasta que ocurre el cambio ya sea este en forma positiva o negativa.

1.1.2 Definición y complejidad de la sociología

La sociología comenzó a desarrollarse como ciencia al final del siglo XIX y todo esto porque la industrialización cambió la forma de las sociedades y la naturaleza de la vida social, la sociología se quedó corta en el intento de entender la revolución industrial y el crecimiento de los modernos estados-nación.

En la actualidad la visión de la sociología ha cambiado a consecuencia de los avances tecnológicos las transformaciones culturales y los trastornos sociales y la meta inmediata es el crear una estructura para entender el rápido cambio social en cuestiones públicas y las vidas privadas.

La sociología como ciencia nace en la Francia revolucionaria La sociedad europeas después de la Revolución Francesa era diferente, tres hechos fundamentales permiten explicar el origen de la sociología y comprender su proyecto y estas son:

Las alternativas en el contexto económico social

Las modificaciones que se producen en el contexto económico social, se deben a la revolución industrial, desde el punto de vista sociológico, los componentes de esa revolución que más atrajeron la atención de los primeros sociólogos fueron:

Los conocimientos científicos se transforman muy rápidamente en nuevas tecnologías que aplicadas al proceso de producción desarrollan prodigiosamente la energía de que dispone cada trabajador y el rendimiento de la fuerza de trabajo, así como la producción industrial reclama concentración de la fuerza de trabajo, al mismo tiempo gracias al carácter científico de la organización del trabajo y a la aplicación de la tecnología al proceso de producción, crece la riqueza global de la comunidad, se multiplican las crisis económicas que producen la existencia de una pobreza desconocida hasta entonces en medio de una riqueza también desconocida, mientras hay millones

de seres humanos que viven miserablemente, hay almacenes de mercancías que no llegan a venderse.

Las modificaciones en el marco político

Las relaciones entre la sociedad civil y el estado también experimentan variaciones importante a partir de la revolución francesa de las ideas democráticas la expansión de la convicción de que todos los ciudadanos son iguales entre sí y que la nación es algo formado por todos y en lo que todos tienen derecho a participar.

La tecnología

Por último hay que referirse al gigantesco avance experimentado por los conocimientos científicos y técnicos desde el renacimiento, estas series de hechos hacen imposible seguir analizando la sociedad y la vida social según los métodos y conceptos heredados cuando se trató de comprender lo que estaba pasando surgió el intento de una nueva teoría la ciencia de la sociedad.

Fue Comte el que inventó la palabra sociología en 1837 hasta ese momento denominada física social y por ello se le considera el padre fundador de la sociología sin embargo muchas de sus ideas las tomó de Saint-simon para quien trabajó como secretario Saint-Simon es de los más lúcidos testigos de la revolución había nacido en 1760 y se da plena cuenta del desorden moral y político en que aún vivía Francia en 1820 explico su visión de la historia pero la ley de los tres estados de comte es mucho más clara y taxativa.

Comte concibe la sociología como conocimiento y acción a la vez creía que la sociología debía de contribuir al bienestar de la humanidad, con la obra de comte la sociología como ciencia esta fundada.

Siguiendo en Francia y avanzando hasta el final del siglo XIX hay que recordad a Emile Durkheim la sociología camina ya sus tres obras primeras son: la División del trabajo social, las reglas del método sociológico y el suicido.

La sociología nace como un intento de comprender la extraña realidad social de que está hecho el hombre, la sociología es el estudio de la vida social, humana de los grupos y sociedades es una empresa cautivadora y atrayente al tener como objeto nuestro propio comportamiento como seres humanos.

La sociología se encuentra dentro de la familia de las ciencias sociales como son la psicología, antropología, economía, ciencias políticas y partes de la historia ya que todas estas tienen que ver con el comportamiento humano.

La sociología se encarga del estudio no solo de grupos sino también en las organizaciones, se estudia las comunidades de grupos pequeños pero también se examinan sociedades industriales modernas y organizaciones a grandes escalas.

Un sociólogo estudia la naturaleza de la acción social en sí misma, la manera en que los diferentes tipos de actividades encajan para crear el gran patrón de la sociedad.

Un sociólogo basa sus estudios en áreas donde la estructura social y la cultura se enlazan como en las instituciones sociales ya que estas son patrones establecidos de acción y en ellas es posible el desarrollo de diferentes actividades sociales como son la familia, la educación, la religión y los sistemas político y económico, con todo esto se colabora para la toma de decisiones del sistema político, sistema económico creando con esto una estructura básica de la vida social.

Una característica especial de la sociología es la estratificación social la cual se refiere a la división de la sociedad en diferentes niveles en los que las personas tienen un acceso desigual a las oportunidades sociales y las recompensas.

La sociología es el estudio sistemático riguroso y científico de la sociedad, la sociología implica un modo particular de ver el mundo que nos rodea, una determinada perspectiva, una de las funciones de la sociología es la de mostrar las diferentes formas en que los arreglos sociales se integran en nuestras vidas, proporcionando un contexto comparativo mediante el cual se valora las condiciones de nuestra sociedad y nuestras propias experiencias personales.

Los sociólogos no solo desafían las creencias populares y buscan mantener el registro exacto separando los hechos de la ficción también investigan los orígenes sociales de esas creencias.

1.1.2 La perspectiva sociológica

La perspectiva sociológica consiste en ver lo general en lo particular, los sociólogos son capaces de identificar de lo general mediante la experiencia social de las personas.

Los sociólogos reconocen que cada ser humano es único pero también es importante definir en la categoría que se encuentran y a la que pertenecen ya que las categorías son las que sirven para clasificar a las personas influenciando las experiencias vitales de esas personas.

Un ejemplo de la perspectiva sociológica en nuestras sociedades esperamos que los niños sean dependientes los adultos responsables y los ancianos que se hagan a un lado esto lo sabemos comparando la evolución de las sociedades dentro de otras sociedades se espera que los niños sean totalmente independientes los adultos no intervengan en su relación con ellos y los ancianos altamente respetados y valorados socialmente.

Los individuos experimentan el poder de la sociedad cuando al hacer sus planes de vida, tienen que tomar en cuenta las oportunidades y desventajas que les imponen según el sexo con el que han nacido.

Ahora bien como utilizar la perspectiva sociológica, es regresando y viendo las cosas no de un modo familiar ya que todos sabemos que muchas de las veces las cosas no son como parecer.

Cuando es un sociólogo el que ve las cosas este cuestiona lo que hacen, piensan, deciden ya que en gran parte esto esta determinado por la sociedad en que viven cada uno de ellos, ya que la sociedad tiene una gran influencia en nuestras acciones y la sociología nos muestra las pautas y los procesos sociales que terminan afectando nuestras acciones y nuestras decisiones.

Una demostración de cómo influye la sociedad misma en nuestras decisiones es cuando se cuestiona a varias personas sobre el porque de los estudios que esta realizando y las contestaciones son porque no tenía otra opción, porque así lo decidieron mis padres, porque ahí están todos mis amigos y en realidad esta decisión esta dada por la influencia de la sociedad ya que años atrás no era muy común que se estudiará en la Universidad y mucho menos si tu sexo era femenino, existían mas oportunidades de trabajo y por lo regular la mujer formaba un hogar a partir de los 18 años, más sin embargo en la actualidad las

cosas han cambiado en forma radical, es necesario el estudio universitario por la competencia en el mercado, la edad aproximada de formar un hogar va entre los 25 y 30 años por lo tanto no son decisiones propias sino que van en relación con lo que sucede en la sociedad.

Existe una demostración de cómo la sociedad influye en nuestras decisiones y conductas es quizá el estudio del suicidio.

Emile Durkheim un pionero de la sociología, eligió el suicidio como tema de investigación y demostró que en el suicidio intervienen determinadas variables sociológicas, Durkheim analizó datos sobre suicidios en Francia y otros países y descubrió que las tasas de suicidio de los hombres, protestantes, las personas con un alto nivel económico y solteros eran mucho mas altas que los católicos y judíos, con escasos recursos económicos y casados y esto es el grado de integración social en que se encuentran.

Durkheim decía que un mayor grado de autonomía personal implica un menor grado de integración social y, así una mayor probabilidad de suicidios y con los estudios realizados y las encuestas es como la sociología va creciendo y comparando unas sociedades con otras y así manteniendo una perspectiva global.

Que importancia tiene la perspectiva global en nuestros tiempos donde la tecnología tiene un tremendo avance, la perspectiva global estudia y analiza de los fenómenos que ocurren a nivel mundial y de la posición que cada sociedad ocupa en relación a otras y dentro del sistema mundial, ya que la perspectiva global es como la continuación de la perspectiva sociológica.

Dentro de la sociedad todo lo que ocurre en esta afecta las decisiones y las experiencias de las personas ya que no podemos vivir aislados, la perspectiva global obedece a tres razones que son las siguientes:

Las sociedades se encuentran vinculadas por la multiplicación de los intercambios comerciales entre las naciones ocasionando con esto una economía global y todo esto esta relacionado con el proceso de la globalización donde los lazos que unen a los países son cada vez mucho mas interconectados.

Ahora bien la perspectiva global nos permite darnos cuenta de que mucho de la problemática que existe en un país existe en otros, la contaminación es un

problema realmente general ya que lo que ocasiona un país repercute en todo el mundo.

Pero cuando utilizamos la perspectiva global hacemos más flexibles nuestros razonamientos acumulamos nuevas herramientas para tomar mejores decisiones y así de esta forma logramos obtener los beneficios de la perspectiva sociológica :

La perspectiva sociológica termina convirtiéndose en una forma de pensar una especie de talante crítico que sirve para poner en cuestión los valores las normas las definiciones de las cosas o incluso la forma de hacer las cosas como es trabajar amar morir que en muchas ocasiones ni nosotros mismos las damos por supuestas, el llegar a pensar en términos sociológicos podemos llegar a la conclusión de que algunas de nuestras ideas que considerábamos naturales o indiscutibles terminan apareciendo dudosas o falsas, así mismo la perspectiva sociológica nos permite conocer mejor las oportunidades y los obstáculos que podemos encontrar en nuestras vidas, mediante la sociología podremos ver como opera la sociedad con ciertas reglas, hace que seamos miembros activos de nuestra sociedad pues cuando no conocemos como opera la sociedad es mucho más fácil que aceptemos las cosas como son pero cuanto mayor sea el conocimiento de otras sociedades mejor podremos entender las instituciones los valores tradiciones etc. De la sociedad en la que vivimos y de esta forma se nos considera como miembros activos.

También la sociología nos ayuda a reconocer que existen diferencias entre las sociedades a reconocer el sufrimiento humano y a afrontar el reto de vivir en un mundo tan complejo y tan plural.

1.1.3 Hechos sociales y causas sociales

La sociología es parte de un mundo en constante transformación que tiene como objeto el estudio de las sociedades y estas estan cambiando constantemente ya que estas no son objetos estáticos y mientras el sociólogo estudia estas se encuentran en pleno cambio por ejemplo un descubrimiento hecho hoy puede que tengamos que cuantificarlo sustancialmente mañana a la vista de nuevos acontecimientos que nos dan pistas hacia los nuevos fenómenos.

Otro problema es el etnocentrismo esto es la tentación de valorar o evaluar otras sociedades no desde una perspectiva global sino desde la propia sociedad ya que los grandes acontecimientos históricos no se producen porque sí son el resultado de determinadas causas sociales complejas y sólo en parte predecibles.

Muchos de nuestros juicios y preferencias, que consideramos muy personales y hasta originales no son en realidad otra cosa que el reflejo de las fuerzas sociales que han obrado sobre nosotros, así cuando usamos calificativos como obra maestra, despreciable, brillante, degenerado y otros semejantes, no sólo estamos expresando juicios personales sino también el influjo de los valores de nuestra cultura en nuestros propios gustos y opiniones.

Cualquier convicción o actitud o comportamiento que comparten las personas en una sociedad, es el producto del grupo social, es un hecho social que viene a ser las propiedades de la vida de grupo que no pueden ser explicados por las acciones, por los sentimientos o por las características de las personas individualmente.

Además de identificar los hechos sociales, los sociólogos también buscan determinar las fuerzas sociales que los producen, los sociólogos buscan explicar los hechos sociales en función de otros fenómenos sociales.

La reflexión sociológica se ve afectada en todos sus niveles por las circuntancias socio-culturales en que se desarrolla, el sociólogo en cuanto tan, forma parte también de una sociedad y una cultura concretas y su reflexión sociológica es también un hecho social esto mismo lleva a plantearnos la relación entre la sociología y los valores.

El sociólogo busca la problemática social por ejemplo violencia familiar hace años atrás se pensaba que la violencia física dentro del hogar se daba en muy pocos casos en la actualidad con tristeza nos damos cuenta que no es así y que además existe algo atrás de todo esto, estudiosos descubrieron que en los sesenta fueron años de violencia la guerra de Vietnam, protestas masivas , asesinatos como el de John F. Kennedy y Robert Kennedy, Martin Luther King Jr., diferencias raciales y se dio un aumento las estadísticas en cuanto a violencia esto en forma mundial, con esto los sociólogos sospecharon que la violencia pública surge de la brutalidad privada ya que la gente aprende a usar la violencia y desgraciadamente la hace suya, es una forma de vivir entonces

la percepción pública acerca de la violencia íntima se transformó de problema privado a un problema social que en la actualidad vivimos mundialmente.

Los hechos sociales y sus causas surgen por las desviaciones que es la violación de las reglas sociales quebrantando los valores y normas de una sociedad o grupo.

Las personas distinguen entre dos amplias categoría de comportamiento lo bueno y lo malo y cada sociedad tiene sus normas y reglamentos y experimenta violación a estas normas y reglas y de esta forma se castiga la delincuente es una desviación universal pero hay que tomar en cuenta que lo bueno y lo malo en muchas ocasiones es relativo dependiendo la cultura, actos que son considerados como sagrados en algunas culturas pueden verse como pecado, sacrilegio o enfermedad en otras.

En problemas que van desde el abuso a la esposa hasta los delitos de cuello blanco, el racismo la desigualdad, la corrupción etc. Los sociólogos están en la búsqueda de soluciones a todo este tipo de hechos sociales y de esta forma poder diseñar programas que se apliquen a la sociedad, pero en muchas ocasiones nos encontramos que dichos programas van en contra del pensamiento político y que además representa un compromiso de tipo personal .

El instrumento más poderoso de control social es la socialización porque esta coerciona a las personas para hacer lo que se supone que deben de hacer pero como no todos somos iguales para lograr el control todas las sociedades dependen de sanciones.

1.1.4 La imaginación sociológica

Entre los objetivos de la sociología es buscar y ayudar a los individuos a que puedan entender sus propias experiencias, proyectos y problemas y se llega a la conclusión de que cada problema y solución esta en manos de cada uno de los individuos, mientras veamos que la historia y la vida individual no estan en forma separada sino forman parte de una sociedad.

Un ejemplo real es la extrema pobreza, la desigualdad social, desempleo, que existe no solo en nuestro país sino en el mundo entero ocasionando uno de los problemas más vistos en la actualidad, es la falta de hogar cada uno tiene su

propia historia de esa falta de hogar pero esto no deja de ser un problema social, siendo el resultado de causas sociales y desgraciadamente no puede ser resulta por actos solitarios de actos de caridad individual.

Que papel juega la imaginación sociológica en esta problemática, esta nos permite ver y comprender que cada fracaso y defecto personal es el resultado de fuerzas sociales y se convierten en patrones sociales.

La violencia familiar es inevitable en una sociedad que enaltece la violencia mediante los medios de comunicación, ahora bien la imaginación sociológica no está limitada a sociedades con niveles bajos de valores y principios sino también se aplica a la gente en circunstancias cómodas.

Mencionare un ejemplo, hace varias décadas la gente empezó a fumar sin conocer las consecuencias tan graves que esto trae y ahora tenemos la oportunidad de recibir mucho mas información sobre esta adicción de hecho los niños la reciben desde que se encuentran en la escuela con esto hemos ganado el derecho a los ambientes libres de fumadores mediante una acción colectiva.

La imaginación sociológica no puede prometer soluciones pero lo que si permite es que por medio de la nueva información y con relación a lo que se vivía años atrás tenemos la oportunidad de planear, organizar y diseñar programas para un cambio de vida y podernos adaptar al mundo en que vivimos utilizando el sentido común y los medios informativos.

El sentido común, el cual muchas veces se dice que es parte de la sociología porque los sociólogos están interesados en temas que corresponden a todos y en ocasiones los descubrimientos sociológicos y el sentido común.

La sociología como el sentido común a veces se involucra con lo obvio y mundano la sociología pone lo obvio a prueba esta es la diferencia sustancial entre la sociología y el sentido común. La mejor forma de ilustrar esta diferencia es asumir una perspectiva fresca del sentido común ya que los sentidos son la base.

En último término, la reflexión sobre nuestra experiencia no puede ser otra cosa que sociológica porque la sustancia de nuestra vida está hecha de relaciones con otros. Relacionar lo que nos ocurre personalmente con lo que sucede a nuestro alrededor es lo que Wright Mills llamó imaginación

sociológica, algo así como el contrapunto frío a nuestras fantasías sin la imaginación sociológica no resulta fácil entender los hitos de nuestra biografía.
La humanidad ha tenido que enfriar mucho sus emociones y sus creencias, sus mitos y sus ensoñaciones para llegar a ese descubrimiento. En el camino hacia tan sencilla y con frecuencia desoladora percepción se han quedado las explicaciones que nos daban o nos dábamos sobre nuestros encuentros y desencuentros, no encuentras trabajo porque eres un vago, porque no buscas suficientemente, porque no aceptas lo que te ofrecen, es el epíteto que tanta gente joven recibe hoy cuando el mercado de empleo no les es propicio y menos mal que esta condena no suele ir acompañada de otra más rotunda de tiempos pasados, no encuentras trabajo porque Dios quiere castigarte por tu indolencia.
Entender las circunstancias del mercado de empleo, emplear la imaginación sociológica, puede que no haga más llevadero el trance pero por lo menos, sirve para uqe los desempleados no acepten el insulto añadido y se desanimen aún más.
La imaginación sociológica no ha solido estar en el currículo escolar, aunque desde hace cincuenta años en Europa y algo menos en España, los estudiantes de secundaria reciben clases de ciencias sociales y de una historia algo más contemporánea. Claro que las otras pedagogías, la calle los amigos lo que uno lee o escucha y la reflexión resultante equipan a las nuevas generaciones para entender las causalidades sociales casi con la misma claridad con la que las ciencias más duras les ayudan a entender las otras casualidades.

Los niños pobres adquieren la imaginación sociológica antes que los niños ricos y, sobre todo, que los de la clase media tradicional que, hace cincuenta o sesenta años, eran exhortados por las pedagogías formales y las informales a evitar un pronto encuentro con la dura realidad y a enmascarar ésta. La literatura picaresca existente en todas las lenguas y culturas nos presenta un variado muestrario de gentes hábiles en la gestión del a pobreza astutos en las relaciones humanas, lo suficientemente al menos como para averiguar los propósitos que los más poderosos tenían sobre sus vidas y tratar de evitarlos.

También los refraneros populares, en sus versiones ora y escrita contienen ese recetario del que el pueblo ha echado mano para entender al mundo

La imaginación sociológica es como el currículo invisible de nuestro aprendizaje vita, una maduración progresiva que nos ayuda a defendernos de los poderes y sus apéndices ideológicos tratan no solo de gobernar la realidad sino, sobre todo de imponer su interpretación sobre ella el rey gobierna en nombre de Dios el sexo sólo es legítimo dentro del matrimonio.

De la imaginación sociológica vamos echando mano a medida que maduramos a la vida y aunque ésta es más sabrosa cuando no la analizamos constantemente puede resultar desastrosa si prescindimos de ella.

1.2. La sociología como ciencia

El conocimiento científico es un conocimiento que se adquirió metódicamente y está sistemáticamente organizado el cual consiste en un cuerpo de verdades generales o por lo menos lo contiene, es objetivo en el sentido de que todo observador sin prejuicios debería admitirlo si se le presentan las pruebas.

Además que nos permite hacer predicciones acertadas y en consecuencia, controlar el futuro de los acontecimientos, en alguna medida al menos.

La sociología ha tratado de buscar la evidencia y de identificar las condiciones bajo las cuales resultan verdaderas sus afirmaciones específicas, la sociología ha avanzado porque los sociólogos han estado convencidos de que el método científico puede contribuir grandemente a nuestra comprensión del carácter del hombre, sus actos y las instituciones así como a la solución de los problemas prácticos a que se enfrentan los hombres en sus vidas colectivas.

El esfuerzo sistematizador ha sido también una herencia de la sociología desde su fundador Augusto Comte, que nos presenta su proyecto globalizador, hasta los teóricos actuales que se esfuerzan por presentar visiones complexitas de la realidad social.

La sociología pretende llegar a generalizaciones y no sólo estudiar hechos o entidades individuales. Como todas las otras ciencias, se basa en el supuesto de que hay un orden natural que el hombre puede descubrir.

La sociología como conocimiento acerca de la vida social cumple todos los requisitos del conocimiento científico.

Antes de continuar es importante definir que es una ciencia es un conocimiento racional, sistemático, exacto, verificable y por consiguiente falible y distingue entre las ciencias formales que son las que se ocupan de inventar entes formales y de establecer relaciones entre ellos y las ciencias fácticas, empíricas o materiales que son las que se refieren a sucesos y procesos.
Estas ciencias pueden ser de la naturaleza física, química etc., de la cultura sociología, psicología, derecho, etc.

Los rasgos esenciales de las ciencias son:

Racionalidad es un conocimiento constituido por conceptos, juicios y raciocinios, un conocimiento cuyas ideas pueden combinarse de acuerdo con algún conjunto de reglas lógicas, con el fin de producir nuevas ideas, un conocimiento cuyas ideas se organizan en sistemas de ideas, esto es, conjuntos ordenados de proposiciones.
Objetividad que el conocimiento científico de la realidad es objetivo significa, que concuerda aproximadamente con su objeto, es decir, que busca alcanzar la verdad fáctica, que verifica la adaptación de las ideas a los hechos recurriendo a un comercio peculiar con los hechos y el intercambio que es controlable y hasta cierto punto reproducible.
Características de la ciencia fáctica, el conocimiento científico trasciende los hechos, es claro y preciso, es comunicable, verificable, metódico, sistemático, general, legal, predictivo, la ciencia es analítica, es explicativa, abierta y útil.

Como conocimiento científico diremos que es un conocimiento que se adquirió metódicamente y está sistemáticamente organizado, consiste en un cuerpo de verdades generales o por lo menos lo contiene, es objetivo en el sentido de que todo observador sin prejuicios debería admitirlo si se le presentan las pruebas, nos permite hacer predicciones acertadas y en consecuencia controlar el futuro de los acontecimientos, en alguna medida, al menos.
La sociología como conocimiento de la vida social cumple todos esos requisitos, la sociología ha tratado de buscar la evidencia y de identificar las condiciones bajo las cuales resultan verdaderas sus afirmaciones específicas.

La sociología ha tratado de buscar la evidencia y de identificar las condiciones bajo las cuales resultan verdaderas sus afirmaciones específicas.

La sociología ha avanzado porque los sociólogos han estado convencidos de que el método científico puede contribuir grandemente a nuestra comprensión del carácter del hombre, sus actos y las instituciones, así como a la solución de los problemas prácticos a que se enfrentan los hombres en sus vidas colectivas.

El esfuerzo sistematizador ha sido también una herencia de la sociología desde su fundador Augusto Comte, que nos presenta su proyecto globalizador, hasta los teóricos actuales que se esfuerzan por presentar visiones compelxivas de la realidad social.

La sociología pretende llegar a generalizaciones y no sólo estudiar hechos o entidades individuales. Como todas las otras ciencias, se basa en el supuesto de que hay un orden natural, que el hombre puede descubrir.

Si realmente no hubiera tal supuesto, si no existiera tal orden no podría haber ciencia.

La introducción de este supuesto en el estudio del hombre y la sociedad es responsable del desarrollo de la ciencia social.

Las generalizaciones científicas deben ser sometidas, directa o indirectamente, a las pruebas empíricas, por muy lógicas o razonables que sean las generalizaciones contenidas en una teoría, no tendrán nivel científico a menos que sean confirmadas por una evidencia fundada.

Los meros hechos no pueden hablar por sí mismos sólo cuando están vinculados con ideas generales pueden ser incorporados a un conjunto de conocimientos científicos.

Un ejemplo de esto lo vemos en el retroceso continuo y relativamente rápido del analfabetismo en México adquiere significado sociológico sólo cuando ese retroceso se vincula con el estado de la economía, con los valores de la sociedad y con otros rasgos de la vida mexicana contemporánea.

Es probable que la objetividad sea más difícil de lograr en las ciencias sociales que en las naturales, por ello será normal que los sociólogos introduzcan en sus investigaciones todo un conjunto de concepciones que deben eliminar o

controlar a fin de evitar que sus hallazgos contengan observaciones prejuiciados o interpretaciones personales que los alejen de la objetividad.

Son muchos los factores que contribuyen al desacuerdo y a la falta de objetividad entre los científicos sociales, las tendencias personales, los prejuicios de grupo, las teorías antagónicas de interpretación de los hechos, los conflictos filosóficos.

Una de las mayores dificultades a las que se enfrenta la sociología es el hecho de tener que buscar conocimientos seguros en un campo donde todo mundo cree que ya conoce las respuestas, que por lo tanto ve a la sociología como una ciencia de lo obvio.

Aparentemente, todas las conclusiones de la sociología no son sino explicitaciones del sentido común, sin embargo aún en el caso de que eso fuera verdad, la sociología nos prestaría un gran servicio al comprobar científicamente lo acertado o equivocado de las opiniones convencionales, pues sabemos que el sentido común no sólo es frecuentemente defectuoso debido a su imprecisión y sus limitaciones, sino que generalmente ha resultado ser el enemigo mas poderoso del avance de los conocimientos científicos por lo tanto en esta situación, la sociología constituye obviamente una fuente útil e incluso esencial de conocimiento seguro tanto para el individuo como para la sociedad .

1.2.1. La ciencia de la sociología

La sociología es el estudio científico de la sociedad, las relaciones humanas, el comportamiento humano en grupo y los problemas sociales, en cualquiera de esos aspectos que consideremos, la sociología significa el estudio sistemático de los siguientes hechos, el carácter social del hombre, la necesidad que tiene el ser humano de vivir en sociedad para satisfacer sus necesidades básicas, las diferentes agrupaciones de que forma parte el hombre al vivir en sociedad, las distintas formas en que se relacionan los hombres dentro de una comunidad.

La ciencia de la sociología trata diferentes problemáticas, la ciencia es un conjunto de procedimientos acordados para establecer y explicar hechos, la sociología comparte cinco características clave con otras ciencias.

La sociología es una disciplina empírica que descansa sobre la evidencia recolectando a través de la observación sistemática y la experimentación, ya que los sociólogos trabajan con información verificada mediante la observación, demandando pruebas.
Por supuesto que la sociología toma en consideración los presentimientos y la especulación pero de ninguna manera basa sus conclusiones bajo estos términos.

La sociología busca mediante técnicas el bajar el índice de errores y estar en constante verificación de los resultados ya que la sociología es un riesgo público ya que muchas de las veces las conclusiones a las que se llegan no son siempre las absolutas, la sociología tiene un mecanismo de auto corrección, ya que basa sus estudios e investigaciones estudiando de los casos particulares para llegar a generalizaciones.
Mediante todo esto la sociología busca producir teorías, que ayuden a predecir entender y explicar eventos.
El estudio científico de los grupos y sociedades a logrado un sin fin de descubrimientos lo que ha permitido a los sociólogos emplear la lógica y métodos usados por todos los científicos.
La investigación científica es un proceso creativo requiere imaginación, tolerancia, para el trabajo duro y algunas veces tedioso habilidad para organizar recursos y cooperación activa y ocasionalmente suerte.

1.2.3. La sociología y otras ciencias

La sociología no es una ciencia exclusiva del estudio de todas las relaciones humanas dentro de la sociedad, existen ciencias directas y ciencias laterales a la sociología , las ciencias directas son :

La economía, esta busca que se cumplan las necesidades primarias y secundarias, el producir y el repartir los bienes y servicios.

La historia mediante esta conocemos el pasado del hombre y de esta forma su comportamiento .

La antropología esta estudia al hombre con su entorno.

La Psicología ayuda al estudio del comportamiento del ser humano.

Como ciencias laterales encontramos a la antropología social, filosofía social psicología social, ética social

Pero La ciencia no solo pide hechos sino que exige descripciones minuciosas y por tal razón es importante señalar que siendo la sociología el estudio de la sociedad como un conjunto de hombres en interdependencia, es importante señalar la relación que existe entre la sociología y las demás ciencias que estudian a los hombres como individuos o como grupos .

Empezaremos por la anatomía y la fisiología estas estudian la estructura y funcionamiento de los seres humanos que se repiten en todos los hombres.

La antropología física estudia la variabilidad de la estructura corporal de esos seres y los clasifica en variedades formando grupos nominales o estadísticos de individuos que presentan rasgos distintivos hereditarios y externos similares.

La psicología estudia los procesos mentales que tienen lugar en las mentes individuales y nos dice cómo ve, oye, siente reacciona a los estímulos etc., un ser humano.

Si podríamos decir que la sociología en sí no estudia ni se interesa por lo que ocurre con los procesos del hombre ni su funcionamiento pero si se interesa lo que ocurre cuando los hombres se reúnen formados ya sea en masas o grupos, cuando luchan se imitan cuando se desarrollan entre sí para formar una cultura sus valores sus mitos y creencias.

La filosofía social esta es considerada como una disciplina mucho más antigua que la sociología, la filosofía social floreció en el siglo XVIII, siendo esta al igual que la sociología la búsqueda de describir y explicar la realidad ya que las dos se basan en la observación de hechos y en las generalizaciones derivadas de esa observación.

La filosofía es primordialmente un intento de comprender la realidad en su conjunto pero es importante señalar que existe una clara diferencia entre la filosofía social y la sociología.

La historia ciencia que trata de comprender a los hombres en interdependencia mas exactamente en configuraciones pasadas de esa interdependencia, la diferencia entre una y otras es que la historia estudia el pasado humano como una sucesión de acontecimientos concretos y únicos de situaciones de procesos y la diferencia estriba en que el historiador muestra lo variable y el sociólogo señala lo constante y recurrente la historia describe la multitud de combinaciones concretas en que se han encontrado los hombres en diferentes situaciones y la sociología descompone las diferentes combinaciones en elementos básicos para de esta forma formular leyes que gobiernen .

Finalmente es necesario diferencias entre la sociología de las ciencias sociales concretas como son la economía, la política y la etnología .

Comte creía que la sociología debía apoderarse de todos los datos estudiados por esas ciencias concretas y unificarlos y de esta forma los privaba de su razón de ser.

Herbert Spencer pensaba que la sociología era una supeciencia que no observaba por sí misma los fenómenos sociales sino que unificaba las observaciones y generalizaciones hechas por las otras ciencias sociales.

La etnología se limitaba al estudio descriptivo en gran parte de las sociedades sin escritura esto ha llevado a que tenga relación con la sociología ya que las dos tienen interés en común.

1.2.3. Niveles de análisis sociológicos

El análisis es la forma de desmenuzar una situación para encontrar el porque, cuando, como, para que, etc.

En sociología encontramos que existen dos tipos de análisis el micronivel y el macronivel .

El micronivel este se encarga de llevar a cabo el análisis en situaciones meramente cotidianas, es como una lente de acercamiento que produce análisis detallados de lo que las personas hacen, dicen y piensan en el flujo real de experiencia momentánea, en este caso las relaciones entre dos

individuos dentro de la organización o del comportamiento de los grupos informales dentro de estas, ya que la estructura de grupo tiene incidencia significativa en los individuos respecto a su manera de actuar y de pensar, sin conocer a los individuos particulares que componen un grupo, es posible predecir patrones regulares en el surgimiento de líderes, respuestas a presión y toma de decisiones.

Un ejemplo sería el siguiente los alumnos actúan de determinada manera dependiendo sus roles que desempeñan esto trae resultados diferentes de la misma persona un estudiante actúa hacia sus profesores en forma diferente de cómo actúa con sus padres, en este caso la relación profesor estudiante es una relación formal jerárquica en la cual la persona tiene más poder y autoridad que la otra, en otra situación la amistad es una relación informal pero jerárquica como resultado diferentes reglas se aplican en cada caso.

Podríamos hacer notar la diferencia entre el micro y macro nivel, en el micro nos referimos a situaciones mucho mas personalizadas mientras que el macro nivel es hacia la estructura más sin embargo los dos análisis se complementan para lograr un resultado óptimo de estudio.

El macronivel esta enfocado a una mayoría de las situaciones sociales y los efectos que estas tendrán es por eso importante tocar los puntos de roles y estatus.

Los sociólogos denominan el estatus para indicar la posición en la que se encuentra un individuo.

El estatus funciona como una dirección social, indicandoles a las personas el lugar donde el individuo se puede integrar dentro de la sociedad, porque es importante conocer el estatus al que cada uno de nosotros pertenecemos, porque nos ayuda a prevenir cómo esa persona se comportará hacia los demás.

Existe lo que los sociólogos llaman el estatus social este es adquirido o alcanzado mediante el esfuerzo personal por sus propias actitudes y habilidades, los estatus adquiridos se forman por las fuerzas sociales.

Otros estatus adquiridos son asignados al individuo en el momento de nacer o en alguna de las etapas de su vida por lo tanto el estatus determina la identidad social de una persona.

El rol es la colección de derechos culturalmente definidos, obligaciones y expectativas que acompañan un estatus en un sistema social, un ejemplo de esto sería es desarrollo de un niño necesita ser formado y educado por sus padres y maestros que tengan el tiempo para jugar que puedan asistir a la escuela y así sucesivamente y los padres por su parte tiene la obligación de querer, respetar, formar y educar así como dar a los hijos casa, comida, sustento, así como tomar algunas de las decisiones que muchas de ellas tendrán relevancia en su futuro como que religión practicarán.

Ahora bien el estatus y el rol son elementos opuestos por complementarios uno no podría existir sin el otro, los roles no sólo invocan normas y expectativas sino también proporcionan oportunidades un rol puede ser pensado como un recurso un medio de exigir, negociar y lograr ser miembro y aceptado en la comunidad social, así como ganar el acceso a los recursos sociales culturales y materiales o capitales, los roles y los estatus ponene las bases para las relaciones sociales, antes de interactuar qunque sea en forma breve, evaluamos el estado de la otra persona respecto al nuestro esto a su vez nos proporciona información sobre las expectativas de rol y que tipo de relación podría desarrollarse.

1.2.4 Interrogantes básicas de la sociología

El estudio de la evolución sociocultural nos ha llevado a los largo de la historia a ir descubriendo los cambios sociales que se suceden cuando una sociedad adquiere nuevos conocimientos en particular nuevas tecnologías.

Un ejemplo sería que las sociedades que sólo disponen de una tecnología rudimentaria sólo pueden alimentar a un número muy limitado de personas, que tienen poco control sobre sus propias vidas y por el contrario las sociedades con gran avance tecnológico tienen de entrada una mayor población con mayores conflictos viven personas diferentes en su cultura, valores, creencias, mitos y sus estilos de vida tienen una gran diferencia.

Actualmente vivimos en una cultura de cambio y podríamos decir que es demasiado con lo que contamos eso hace que no nos demos el tiempo de reflexionarlo porque no solo lo importante es ir avanzando sino que ese avance sea para el beneficio de una sociedad y no de unos cuantos o bien que ese

cambio no nos convierta en mejores personas ni con mayores posibilidades y calidad de vida, por tanto habría que cuestionar ese cambio en el que actualmente nos encontramos sumergidos, ese avance que no nos deja meditar ni reflexionar como es los fast food, el fax, el teléfono celular, el cibersexo, corazones artificiales, la cirugía laxer, la reproducción in vitro, la ingeniería genética, la realidad virtual, fibra óptica, misiles inteligentes un holocausto nuclear, estaciones espaciales, cirugía transexual, imágenes vía satélite.

Por supuesto todo esto no solo beneficia a una sociedad también la afecta en todas las esferas de su vida no es que la tecnología determine la sociedad sino que son los miembros de estas los que determinan la sociedad pero que hay con toda la influencia y los medios de comunicación como logran vender ese avance tecnológico, ya que las tecnologías se activan por los individuos entonces llegamos a la idea de que la tecnología es creada por el propio individuo y es el quien decide como y cuando utilizarla y los fines que para este o estos sean los convenientes.

La evolución sociocultural ha traido a lo largo de la historia diferentes resultados dependiendo el tipo de sociedad su período histórico, su tecnología, el tamaño de las sociedades, el tipo de asentamiento, la organización social.

Si bien la tecnología ha mejorado notablemente el nivel de vida de las personas aumentando el nivel de vida de las personas, la producción de bienes y servicios eliminando enfermedades o creando nuevas posibilidades de ocio, ahora bien la tecnología no produce milagros por sí misma un ejemplo vivo es la pobreza extrema que se vive en muchos paises del mundo ya que la misma tecnología es la que ha creado algunos de los problemas que vivimos como la compra venta de armamento nuclear la extinción de los recursos naturales y la pregunta sería puede la humanidad seguir aumentando los niveles de prosperidad material sin causar daños irreversibles al planeta.

De cierto es que los avances tecnológicos han mejorado la vida en muchos aspectos y han logrado que un contacto en las sociedades a nivel mundial, pero la tecnología además de plantear problemas no puede resolver por sí misma muchos y viejos problemas porque esos problemas no son problemas tecnológicos sino sociales y políticos.

Son dos los interrogantes sociológicos básicos en la sociedad
Que mantiene unida a la sociedad
Cuál es la relación entre el individuo y la sociedad

1.2.4.1. Qué mantiene unida a la sociedad?

Marx subrayaba precisamente la dimensión opuesta, la del conflicto social, la dimensión del conflicto social es lo importante para Marx las élites pueden forzar una tregua en el conflicto o demorar su resolución definitiva pero como más y solo puede dejar de haber conflicto cuando la actividad productiva se base en la cooperación y no en la competencia, ni en la explotación.

Los que pertenecen a una sociedad cuentan con elementos que los hacen que puedan compartir pero no quiere decir que por eso compartan todo, antiguamente era mucho más fácil mantener unida a una sociedad ya que toda su orientación iba enfocada en la forma tradicional pero en la actualidad estan unidas por el tejido organizativo que está en su base y que implica una orientación mas racional frente al mundo.

Para Durkheim las sociedades preindustriales estaban unidas por lazos de solidaridad mecánica, por una conciencia colectiva fuerte, las sociedades modernas y complejas caracterizadas por una conciencia colectiva más débil o porosa se mantinenen unidas porque sus individuos son interdependientes.

Una sociedad se mantiene unida dependiendo la perspectiva que se llegue a adoptar.

Perspectiva funcional en esta la sociedad está unida por un proceso automático de autorregulación, en el cual muchas partes diferentes negocios, gobiernos, familias etc., desempeñan funciones diferentes evolucionando de tal forma que trabajan juntas en forma integrada, desde luego que surgen conflictos y desorganización, pero esos son períodos temporales de reajuste.

Según el punto de vista funcional, la sociedad tiene una tendencia natural a desarrollarse hacia un estado de integración funcional en le cual sus diferentes parte encajan dentro de una totalidad que opera en forma contínua.

La perspectiva del poder es el ejercicio del poder por parte de aquellos que controlan los recursos importantes, se considera como el factor principal que estructura y mantiene el orden social.

Se considera el conflicto como el estado natural e inevitable de los asuntos sociales en que las diferentes gentes o grupos y organizaciones luchan por ganar la partida.

Desde esta perspectiva la ideología de la libre empresa, a menudo elogiada por los ricos y poderosos, puede ser considerada como una racionalización conveniente para perpetuar sus privilegios económicos.

Claramente, los sociólogos no están de acuerdo sobre qué mantiene unidas a las sociedades más grandes.

1.24.2. Cuál es la relación entre el individuo y la sociedad

Perspectiva estructural subrayan la importancia de la estructura social, la forma en que la gente, los grupos y las instituciones están organizados entre sí, se cree que la organización estructural está allí para influir sobre el comportamiento, o por lo menos para limitar las posibilidades abiertas a la gente.

Según la perspectiva estructural, los gustos o preferencial de la gente pueden explicarse por fuerzas sociales que surgen de la forma en que se organiza la sociedad, estas fuerzas son consideradas como algo externo al individuo, algo más permanente que él, algo muy difícil de cambiar por el individuo.

Las fuerzas que se originan en la estructura externa de la sociedad permanenete limitan nuestras preferencias y hacen que pensemos o actuemos en forma predecible.

La perspectiva de acción la acción está creando permanentemente los indicadores sociales que constituyen la sociedad, es un error mirar la sociedad como algo estrictamente externo a las personas, la sociedad siempre esta configurada por las acciones de los individuos.

El comportamiento humano es un proceso creativo basado en la forma como las personas interpretan y reorientan las fuerzas sociales.

Según esto, una tarea central de la sociología es la interpretación de las creencias subjetivas y de las expectativas en las que se basa la acción del individuo.

Todos los humanos son iguales de alguna manera pero diferentes en otras, explicar las similitudes y diferencias entre los individuos es una meta mayor de la ciencia social, se considera que los psicólogos se enfocan en las diferencias individuales y los sociólogos están más interesados en los orígenes de las diferencias colectivas y la pregunta es por qué las personas criadas en diversas culturas o miembros de diferentes clases sociales o los hombres y mujeres se comportan y piensan diferente.

En relación a esto ha ocupado por los científicos más de un siglo y todavía hay quien argumenta que el comportamiento individual y social es producto de la herencia o la naturaleza, según esta idea el tipo de persona que somos es genéticamente preordenado y el drama social humano sigue un guión genético predeterminado por otro lado otros autores sostienen que el comportamiento individual y social es el producto de las experiencias y el aprendizaje o nutrición según esta idea el comportamiento depende del entorno social y la educación por tanto el guión social es parte de nuestra propia fabricación.

1.2.5. Objeto de la sociología

Existe cierta ambigüedad en las imágenes que de ello se ha formado la gente, si preguntaremos a los estudiantes aún no graduados por qué se están especializando en sociología recibimos la respuesta " porque me gusta trabajar con la gente" si seguimos preguntando a estos estudiantes respecto al futuro de su ocupación, a menudo escuchamos que se proponen participar en el trabajo o acción social, todas las respuestas indican que el estudiante en cuestión preferiría tratar con gente que con cosas.

La imagen de un sociólogo implicada aquí podría describirse como una persona interesada profesionalmente es actividades edificantes a favor del individuo y de toda la comunidad, ya que la sociología no es una práctica, sino un intento por comprender.

Esta concepción de la actividad sociológica se encuentra implícita en la afirmación clásica de Max Weber una de las figuras más importantes en el

desarrollo de este campo, en el sentido de que la sociología esta exenta de valores, evidentemente la declaración de Weber no significa que el sociólogo no tenga o no deba tener valores, normalmente el sociólogo poseerá tantos valores como un ciudadano, pero dentro de los límites de sus actividades como sociólogo, existe únicamente un valor fundamental, el de la integridad científica, por supuesto incluso en este respecto el sociólogo como ser humano tendrá que tener en cuenta sus convicciones, sus emociones y prejuicios.

Otra imagen del sociólogo es la de reformar social, Augusto Comte, el filósofo francés de principios desiglo XIX que inventó el nombre de la disciplina, pensaba que la sociología era la doctrina del progreso, el sociólogo desempeña el papel de árbitro de todas las ramas del saber para el bienestar del ser humano.

Si todas estas imágenes del sociólogo suponen a su respecto un elemento de retraso cultural, podemos pasar ahora a algunas otras imágenes de fecha más recinete y atribuirlas a los desarrollos actuales e la disciplina, una de estas imágenes es la del sociólogo como recolector de estadísticas acerca de la conducta humana.

Siendo cierto que una parte considerable de la labor sociológica en este país consiste aun en estudios insignificantes de fragmentos oscuros de la vida social, irrelevantes para cualquier interés teórico más amplio. Una mirada la índice de las principales revistas sociológicas, o a la lista de disertaciones leídas en las convenciones sociológicas, confirmara esta afirmación.

Otra imagen de sociólogo bastante común en la actualidad y relacionada muy estrechamente con la del estadístico, es la que lo concibe como un hombre interesado principalmente en el desarrollo de una metodología científica que puede imponer después a los fenómenos humanos, esta es la imagen que guardan frecuentemente los humanistas y que se presenta como prueba de que la sociología es una forma de barbarie intelectual.

El sociólogo se convierte en un hombre que se designa a sí mismo como superior, manteniéndose apartado de la cálida vitalidad de la existencia común, encontrando su satisfacción no en vivir, sino en valorar fríamente las vidas de los demás, archivándolas en pequeñas categorías y, por lo mismo, pasando por alto posiblemente el significado real de lo que observa.

El sociólogo es una persona que se nteresa por comprender la sociedad de una manera disciplinada, esto significa que él lo descubre y dice acerca de los fenómenos sociales que estudia.

Como científico el sociólogo trata de ser objetivo, procura controlar sus preferencias y prejuicios personales y percibir claramente en lugar de juzgar de acuerdo con una pauta, no pretende que su marco de referencia sea el único dentro del cual puede considerarse a la sociedad, con el fin de comprender la sociedad, o la parte de ella que esté estudiando en ese momento, el sociólogo se valdrá de muchos medios, entre estos se encuentran las técnicas estadísticas, tendrán que preocuparse por el significado de los términos que emplea.

El interés del sociólogo es primordialmente teórico, o sea que está interesado en comprender por su propio bien, es una persona que se interesa intensa, incesante y descaradamente por las acciones de los hombres, se ocupará de cuestiones que otros consideran demasiados importantes para investigarlas de manera desapasionada, claro para otros puede encontrarse demasiado aburrida como por ejemplo la interacción humana que acompaña a la guerra o a los grandes descubrimientos intelectuales, pero también en las relaciones que existen entre unos y otros.

El sociólogo se mueve en el mundo común de los hombres, muy cerca de o que la mayoría de ellos llamaría real.

Desarrollo histórico de la sociología

La sociología es una ciencia moderna, se remonta solo a mediados del siglo XIX, cuatro grandes cambios en la era moderna contribuyeron al nacimiento de la sociología:

La transformación de las filosofías y políticas del gobierno

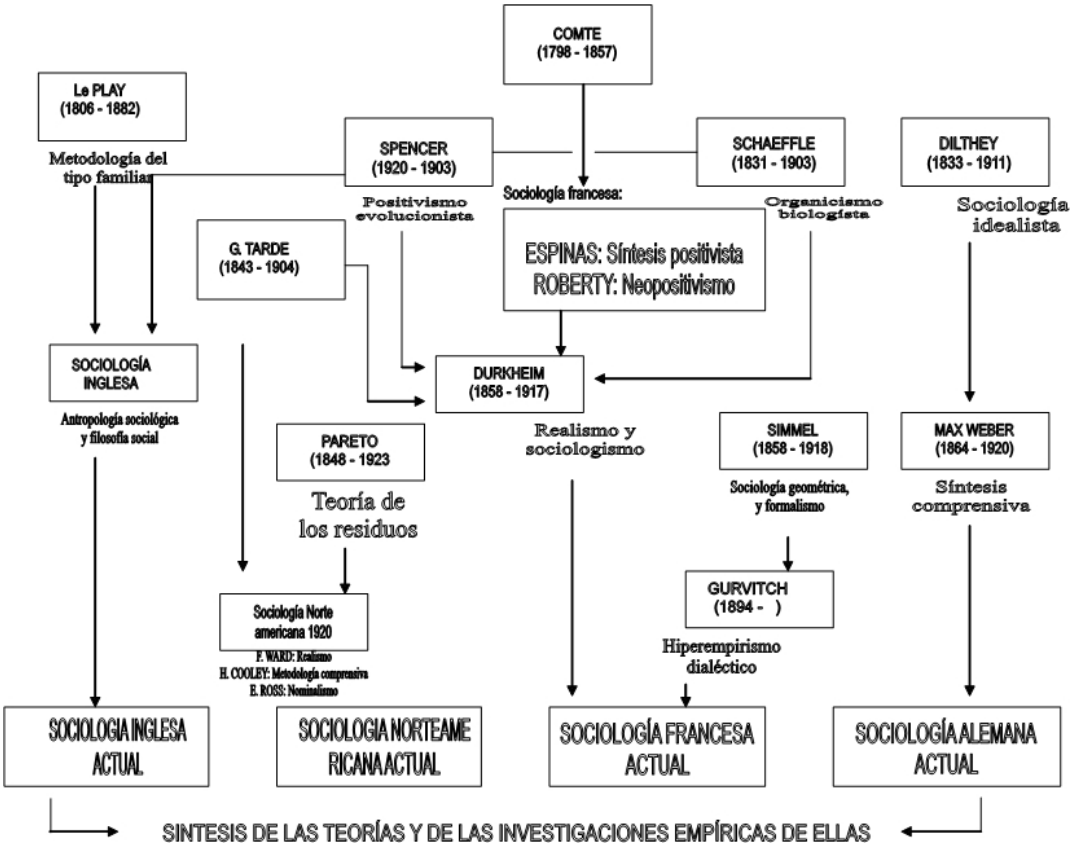
Las difíciles transformaciones económicas que se daban simultáneamente

La amplia diversidad en las costumbres humanas y los valores alrededor del mundo

Las nuevas formas en que la gente empezaba a reflexionar sobre sí misma y su mundo.

Esto ha traído como consecuencia la necesidad de la sociología ya que cada vez más seres humanos se consideraron a sí mismo como racionales que podían aplicar los métodos sistemáticos de la ciencia al ordenamiento y comprensión de sus vidas.

La gente tuvo que admitir que las ideas del sentido común acerca del mundo social resultaban inadecuadas, hacia falta un cuerpo de información sobre los hechos puesto en perspectiva mediante teorías acerca de la sociedad sistemáticamente verificadas.



1.3.1. Orígenes y evolución de la sociología

A lo largo de la historia ciertas personas han sido respetadas por su conocimiento de las costumbres sociales y de la forma como operan las relaciones humanas.

Desde los tiempos históricos más remotos y hasta la época actual, el hombre se ha interesado por la comprensión de la vida social, por estudiar los procesos que en esa vida social se desarrollan y por explicarlos y en lo posible influir de alguna manera en ellos.

Platón (428-347) a.C. en sus diálogos trató de comprender las motivaciones y la conducta de sus contemporáneos y nos presentó sus ideas acerca de la manera como debería funcionar una sociedad ideal.

Algo semejante hizo Aristósteles (384-322)a. C. Quien en su ética nicomaquea nos presenta una filosofía de las cosas humnas según la cual el fin último del a vida humana la felicidad, se define como la actividad del alma según su virtud mas perfecta y mas completa.

San Agustín de Hipona (354-430) en el siglo IV de nuestra era, nos presenta en la ciudad de dios una explicación de la doble naturaleza del hombre su inclinación hacia el bien o el mal, hacia el espíritu o la carne, hacia la voluntad de devoción o la de autoafirmación destructora, hacia el amor y la paz o hacia el odio.

Durante la edad media Santo Tomás (1225-1274) y otros ilustres pensadores reflexionaron sobre la conducta humana y nos legaron un sistema ético y explicativo de la conducta humana basado en la doctrina del cristinanismo

Ya en la época moderna, nos encontramos con obras como el Principe de Maquiavelo y el espíritu de las leyes de Montesquieu en las que se da una reflexión muy seria sobre el funcionamiento del as sociedades, la mejor manera de gobernarlas y la significación de sus normas.

Jacques Turgor (1727 1781) desarrolló la idea de progreso en su discurso sobre la historia universal pasando por tres etapas el hombre suponía que los fenómenos naturales eran producidos por seres inteligentes invisibles pero parecidos entre sí, después viene la explicación de estos hombres y por último observando la acción recíproca mecánica de los cuerpos .

Y no fue sino a partir del siglo XIX cuando comenzó a hablarse del estudio científico de la sociedad humana. Augusto Comte filósofo francés que vivió de

1798 a 1857 acuño la palabra misma de sociología como el estudio de las sociedades y elaboró todo un proyecto minucioso para el estudio científico de la sociedad, a partir de entonces se ha ido desarrollando esta ciencia hasta obtener un lugar respetable entre las ciencias humanas de nuestra época.

La evolución de ciencias que en su mayoría surgieron como disciplinas académicas a fines del siglo XIX tales como la antropología y la arqueología se caracteriza por una sucesión de acercamientos y distanciamientos debidos a diferentes concepciones del ámbito y la metodología de cada una de ellas, con la complicación que implicó diferentes aproximaciones en Europa y Norteamérica.

Ambas fueron tomando forma como uno de los resultados de la expansión colonialista de las grandes potencias occidentales y ese origen explica muchos de los prejuicios e interpretaciones subjetivas que las caracterizaron en sus orígenes, a medida que los investigadores intentaban comprender otras culturas.

Auguste Comte, fue la primera figura importante que sostuvo y demostró con hechos la ciencia de la sociedad tanto en forma empírica como teórica .

Las ideas son las que dan por ende el mundo de las sociedades y por supuesto estas ideas tienden a agruparse y en ellas cada uno de los intelectuales.

A la edad de 16 años ingresa a la escuela politécnica y se intereso en especial por los efectos destructores de la Revolución Francesa ante el desorden creado por la destrucción violenta de grupos sociales intermedios entre la familia y el estado y de esta forma el mejoramiento de la sociedad se convirtió en la principal preocupación de Comte pero pensó que para esto era necesario una ciencia teórica de la sociedad y se dispuso a crearla y para esto se vio en la necesidad de estudiar ciencias que tenían relación con lo que el deseaba estudiando principalmente la filosofía positivista uniéndose a él el Conde de Saint-Simon, socialista utópico que pensaba que los problemas de la sociedad de su tiempo podían ser resueltos reorganizando la producción económica, despojando a la clase propietaria de los medios de producción, de libertad económica, que era el valor más estimado en aquel tiempo.

Comte elabora la ley de las tres etapas que significa ante todo que cada campo de conocimiento pasa por tres períodos de desarrollo el teológico, metafísico y positivo.

Segundo fundador de la sociología Herbert Spencer de 1820 a 1903 pertenecía a una familia de clase media nunca tuvo la oportunidad de asistir a una escuela ordinaria recibió su educación en el hogar y por breves períodos en pequeñas escuelas particulares su preparación llegó a ser solo en matemáticas cosa sorprendente si se tiene en cuenta que escribió tratados notables de biología y de psicología.
La verdadera base del spencerismo es la teoría de la evolución formulando tres leyes fundamentales la ley de la persistencia de la fuerza, la ley de la indestructibilidad de la materia, la ley de la continuidad del movimiento.

1.3.2. La socialización a través del curso de la vida.

Algunos pensadores del siglo XIX ampliaron la teoría darvinista de la evolución y aplicaron el mismo razonamiento a las diferencias transculturales y a las desigualdades sociales.
Darwinistas sociales determinan que el hecho de que en las sociedades europeas algunas personas eran ricas y prósperas mientras en otros eran pobres y hambrientos esto determina la supervivencia del más apto.
Pero otros científicos sociales de este período se enfocaron en el comportamiento si los humanos son miembros del reino animal entonces el comportamiento humano como el de otros animales debe ser gobernado por conductores biológicos e instintos y se pensaba que la guerra y la violencia eran atribuidas al instinto agresivo el comportamiento de masa la instinto de rebaño un interés en la creación de hogares al instinto de nido el amor a un niño al instinto maternal y así sucesivamente.
El punto cerebral de la perspectiva naturalista era la idea de que la mayoría del comportamiento individual y social se predetermina genéticamente, la socialización es poco más que el adorno, por implicación querer rehabilitar individuos o reformar la sociedad es en esencia fútil.

Como agente de socialización es un individuo grupo u organización que influye en el comportamiento de una persona y el sentido de la persona, ya sea para premiar o castigar el comportamiento que proporcionan instrucciones en reglas y roles sociales o simplemente sirven como un modelo.

Es la familia el agente primario de la socialización en la niñez temprana y es una influencia central durante toda la vida.

Aproximadamente en los años cuarenta, cincuenta y sesenta, la familia ejerció un monopolio cercano en las socialización de la niñez temprana ya que por lo menos los niños pequeños pasaban virtualmente todo el tiempo con sus madres y hermanos. Esto contrasta con sociedades tradicionales en que los abuelos tias tios y primos también eran parte integral de la casa .

Por diferentes razones los niños de los años ochenta y noventa dependen no tanto de la familia inmediata sino de otros parientes o cuidadores y maestros de preescolar, con cada vez más madres trabajando aproximadamente el 60% de los niños menores de 6 años pasan la mayoría de su tiempo con alguien que no es su familiar esto se debe principalmente al tamaño y la composición y circunstancias socioeconómicas de la familia de un niño esto no significa que este desapareciendo la socialización ya que existe evidencia de que las madres trabajadores y los padres divorciados le dedican más tiempo a sus hijos que otros padres.

Los padres socializan con sus hijos dependiendo de los patrones que ellos han seguido en el transcurso de su vida.

Todas las sociedades dividen el curso de la vida en etapas, las etapas del ciclo de vida son institucionalizadas en reglas formales que definen las edades a las que se permite a las personas participar en diferentes instituciones sociales y comprometerse en diversas relaciones sociales, nos exigen que asistamos a la escuela pero no permiten casarse o trabajar hasta la edad de 18 años, no se permite la entrada a un asilo hasta la edad de 65 años.

Este ciclo de vida proporciona a los individuos oportunidades y limitaciones en la acción individual pero todavía utilizamos la edad para organizar y evaluar nuestras vidas, todos tenemos los ojos puestos en el reloj social, ya que es la edad la que condiciona la manera en que nos relacionamos con otras

personas, esperamos que un niño, un adulto joven, y una persona mayor se comporte diferente, ynos comportamos de manera diferente hacia ellos.

Cada etapa del ser humano representa una crisis dentro de las habilidades y necesidades del individuo ya que son confrontadas contra las necesidades y demandas de la sociedad y la forma en que el individuo pueda resolver sus conflictos es como se encuentra hacia la siguiente etapa.

Etapas del ser humano son la niñez, en esta etapa es de suma importancia que los padres y los demás agentes de socialización entiendan el deseo de independencia del niño esto le permitirá desarrollar un sentido de iniciativa.

Etapa de la adolescencia en esta etapa el problema es lograr una identidad ante la confusión de roles durante la adolescencia.

Etapa de la adultez temprana es donde se hace el anclaje en la vida esforzándose por establecerse y tener éxito conseguir una mejor vida para la familia, establecer un hogar de base, escoger un empleo o carrera, un compañero, un grupo de valores.

Etapa de la transición de media vida es volverse mas uno mismo consolidar los logros crear un mejor mundo para las futuras generaciones, recuperar la juventud pérdida y revalorar el sentido de la vida.

http://ericps.ed.uiuc.edu./nccic/nccichome.html

1.3.3 Desviación y control social

La desviación viene a ser una violación de las reglas sociales esto sucede en el momento que se quebranta las reglas normas procedimientos y los valores de una sociedad.

Casos como esto lo vemos en el abuso sexual de niños, que es condenado fuertemente pero así mismo ocurren desviaciones que no es posible lograr percibirlas de forma clara y sencilla que en muchas ocasiones es llamada la mentira blanca.

Desde la perspectiva absolutista la desviación queda en el propio acto, que puede verse como una violación de la ley natural o una transgresión contra los mandamientos de Dios.

Las personas distinguen entre dos amplias categorías de comportamiento la buena y deseable y la mala e indeseable ya que cada sociedad establece normas y reglamentos, experimenta violaciones a esas reglas y de una o otra manera castiga a los delincuentes , la desviación es universal, el comportamiento se ve como desviado cuando las personas se alarman se enojan se sienten ultrajadas o amenazadas por violaciones de lo que consideran correcto y apropiado, la definición social de desviación lo que es correcto o incorrecto laudable o culpable es muy relativa actos que son considerados sagrados en algunas culturas pueden verse como pecado, sacrilegio o enfermedad en otras.

Las normas sociales varían de una cultura a otra el islam prohibe consumir bebidas alcóholicas y en contraste el judaísmo y cristianismo incorporan al vino en sus rituales religiosos.

El control social se refiere a cualquier tipo de esfuerzo por prevenir y/o corregir el comportamiento desviado, el instrumento más poderoso de control social es la socialización, en el sentido de la socialización coerciona a las personas para hacer lo que se supone deben hacer más sin embargo la socialización no es perfecta los humanos no son robots sociales, para lograr el control, todas las sociedades dependen de sanciones, es decir premios por comportamiento admitido y castigos por comportamiento desviado.

Existen dos tipos de controles que son los formales y los informales, los informales son presiones extraoficiales para conformar las normas y valores sociales, son tejidos de manera hermética en la vida cotidiana y no ser perciben de forma consciente como una sonrisa, un ejemplo claro es el chisme, es ejercer presión social sin tener que recurrir a las leyes.

Los controles sociales formales son mecanismos públicos institucionales y codificados para prevenir o corregir el comportamiento desviado como ejemplo a esto es la policía, las cortes y prisiones son responsables de reforzar y hacer cumplir la ley mediante los castigos.

Existen otro tipo de controles como son los negocios que premian a los empleados que se encuentran o exceden en productividad con promociones y aumentos y castigan a quienes no lo hacen con despidos.

Otro ejemplo muy claro son las universidades usan las admisiones, becas, calificaciones pruebas y expulsiones para premiar o penalizar a los estudiantes por su actuación.

Los controles sociales se presentan en diversos dominios al mismo tiempo, la casa, la escuela, el trabajo, en la iglesia, en la calle etc.,

http://alcoweb.com/

1.3.4. Teoría de la elección racional

En un primer momento los diferentes fenómenos de la sociedad eran estudiados junto con la política y la economía, estos primeros planteamientos fueron desarrollados a partir de la teoría de la elección racional, esta teoría plantea que las personas cuando van a tomar decisiones se basan en la relación entre los beneficios que perciben y los costos, la teoría de la elección racional toma en cuenta el papel de las acciones específicas en la configuración de los hechos sociales, desarrollando así la perspectiva de acción en la sociología.

Uno de los fundadores de esta teoría es Adam Smith 1723-1790 quien cree que el pueblo toma decisiones económicas, basados en relaciones entre costo y beneficio y al hacerlo sólo consideran las consecuencias personales, más no en los efectos que esto pueda causar en otros, lo cual según Smith, lleva a que se lleve a cabo la producción de suficiente mercancía para el consumidor y por lo tanto un incremento de la riqueza de la población.

Según Smith, la sociedad es un sistema autorregulable en donde muchas partes diferentes actúan según su interés particular pero se engranan todas juntas a través de las fuerzas del mercado, lo que forma un todo que beneficia al común social.

El filósofo Jeremías Bentham 1748-1832, amplia el concepto de esta teoría, ya que demuestra que las personas están motivadas para obtener satisfacción y evitar el dolor Bentham no comparte la idea de Smith que plantea que todas las decisiones de interés particular que realiza el individuo son de gran beneficio para la sociedad, para Bentham, el bien público puede alcanzarse de mejor

manera por medio de una acción gubernamental muy bien planificada, a la cual denomina, “ la mano invisible “

Esta teoría desarrollada y liderizada por Smith y Bentham, ejerce gran influencia sobre muchos campos de la sociología, ya que se aplica a la forma como se toman decisiones de negocios o como las personas deciden invertir en más educación y hasta en la pareja con la cual se van a casar, pues algunas investigaciones han comprobado que los cálculos racionales influyen sobre la elección del pareja, donde los noviazgos son como un mercado donde la gente investiga sobre el mejor producto, basado en los recursos individuales.

Mediante esta teoría se sostiene que, al tomar las decisiones la gente sopesa los beneficios resultantes de una acción particular contra los costos invertidos. Sólo cuando las personas perciben que los beneficios son mayores que los costos adoptan el comportamiento correspondiente.

Subraya el papel de las acciones específicas en la configuración de los hechos sociales.

1.3.5. La teoría de Karl Marx

Este fue uno de los teóricos importantes dentro de la sociología 1818-1883 quién ejerce una importante influencia sobre el pensamiento sociológico y económico, además deser reconocido por haber sentado las bases del comunismo moderno.

Marx cree que los más importante dentro de la sociedad industrial que se desarrolló en su época, era el capitalismo, donde los medios de producción eran propiedad privada y eran utilizados para maximizar las utilidades según Marx, este sistema económico estructuraba todos los demás aspectos de la vida social, creando un conflicto permanente sobre los valores y las metas sociales.

Los intereses de los capitalistas y el proletariado, según Marx, eran contradictorios, pues el mercado competitivo obligaba a los capitalistas a reducir los salarios lo mas bbajo posible para poder obtener mucha más ganancia y por otro lado, los trbajadores se interesaban cada vez más por lograr que el sistema capitalista fuera dejado de lado para introducir un sistema

social donde no existieran clases y en donde la riqueza se distribuyera equitativamente.

Para este teórico, una revolución de trabajadores no podía llevarse a cabo sin una conciencia de clase que presentará intereses compartidos y compromisos fijados, por lo tanto los capitalistas continuaban teniendo el poder en sus manos sobre todos los intereses de la población, creando así una falsa conciencia, lo que hace más difícil el poder llevar a cabo una toma de decisiones más segura y específica. Todo esto hace que el capitalismo se consolide cada vez más.

De acuerdo a lo anteriormente planteado, Marx propone que la sociedad se mantiene unida debido a la destreza de los capitalistas para dominar a los trabajadores, por lo tanto, el poder es la base de la teoría de Carlos Marx, esta teoría toma en cuenta la estructura y la acción social, es estructural en el sentido en que considera las circunstancias históricas del capitalismo como algo que limita la mayor parte de las opciones abiertas al pueblo y es orientada a la acción en el sentido en que reconoce la capacidad de los trabajadores para unirse en la lucha de clases y para cambiar colectivamente las condiciones existentes.

Según la teoría de Marx solo existen dos clases de hombres los explotadores que son los dueños de los medios de producción y los explotados que solo poseen la fuerza de sus brazos.

1.3.6 La teoría de Emile Durkheim

Un importantísimo sociólogo que ha ejercido gran influencia, es el Francés Emile Durkheim de 1858-1917, quien desarrolla la definición de solidaridad social como formas sociales ocultas que hacer posible la unión de las personas, la cual puede presentarse a través de dos formas básicas, una de ellas es la solidaridad une a sociedades pequeñas y sencillas, cuyos integrantes poseen una visión compartida sobre el mundo y desarrollan actividades similares.

Otra parte, la llamada solidaridad orgánica, es aquella que une sociedades grandes, modernas y complejas. En estas sociedades, cada persona percibe ingresos a partir de una ocupación específica y luego utiliza ese dinero para

adquirir mercancías y servicios, que han sido producidos a su vez por otras personas que desarrollan un rol especializado, la relación en esta sociedad se presentan porque las diferencias en sus habilidades hacen que se necesiten para vivir.

Durkheim pensaba que el estudio de la sociedad es muy diferente al estudio del individuo, su teoría fue básicamente funcional, pués para él toda la sociedad se mantiene unida debido al trabajo o función que desarrolla cada individuo dentro de ella.

Este teórico también presenta una perspectiva estructural, pues cree que la sociedad era algo externo al individuo que limita sus compartimientos y resiste los esfuerzos del pueblo por un cambio social.

Para Durkheim, los orígenes de un hecho social, conocido este como propiedades de la vida social que no tienen explicación y que surgen de la interacción social, son en cierta forma menos importantes que la función desempeñada por el mismo en el orden social. Según Durkheim, el crimen y la desviación en pequeñas cantidades, pueden ser socialmente útiles, pues hacen que la gente comience a tener conciencia de la moral que predomina y poder condenar aquellos que no la cumplen. Según Durkheim se crea un estado de anomia, que se conoce como la inestabilidad social y la pérdida de fe en las reglas sociales y las institucionales como resultado del rápido cambio social.

1.3.7 La teoría de Marx Weber

Según Marx Weber 1864-1920 las explicaciones sociológicas se derivan de la compresión de por que el pueblo elige ciertas acciones, creencias que difiere enormemente del planteamiento de Durkheim que habla de la separación del estudio de la sociedad y el individuo, Weber cree que hay aspectos sociales que deben ser estudiados a través de métodos científicos, pero también plantea que los hechos sociales son el resultado acumulado de acciones individuales.

El estudio de Weber se centra en la observación e interpretación de las creencias subjetivas de las personas, sus actitudes, valores y motivaciones. Weber también deseaba entender al igual que Marx y Durkheim, los rápidos cambios sociales que ocurrían en su época Según su época, según este

teórico la historia de la sociedad occidental se basa en un cambio de ls orientaciones tradicionales hacia orientaciones mas racionales.

Esta racionalización puede observarse, según Weber en muchos aspectos de la vida social, como por ejemplo en el desarrollo de la ciencia como principal método para la adquisición de conocimiento, otro ejemplo de racionalización puede observarse en el surgimiento de la burocracía y otro es el desarrollo del capitalismo.

El enfoque más común de Weber es el de las creencias subjetivas y los valores ue conducen a formas particulares de acción y a las acciones que a su vez producen cieros modelos sociales más amplios. Weber utilizó el modelo denominado tipo ideal como instrumento para analizar algunos fenómenos como por ejemplo el capitalismo este modelo no se encarga de mostrar lo que es ni tampoco lo que debiera ser, es mas bien una abstracción que permite al sociólogo observar ciertas características de determinado fenómeno.

Una de las ideas centrales en los planteamientos de Weber es creer que la sociedad es el producto de las acciones de los individuos al igual que Marx, Weber proporciono una teoría sobre lo que hace posible la unión de la sociedad, la cual planteaba que el pueblo lucha por el poder para determinar la naturaleza de sus mecanismo sociales, incluyendo a los gobiernos y agregaba que las agrupaciones políticas y opiniones del pueblo no siempre reflejan sus intereses económicos, para Weber el ejercicio de poder es fundamental ara el orden social.

1.3.8. Otros autores del pensamiento sociológico.

El pensamiento de Montesquieu el origen de la sociedad , leyes naturales positivas , tipología político social, la causalidad y el espíritu general .

Rousseau 1712 1778 el individuo y sociedad, el estado de naturaleza.

Saint-simon 1760 1825 industrialismo y utopía

Augusto Comte 1798-1857 el advenimiento de la filosofía positiva, la ley de los tres estados.

Herbert Spencer 1820-1903 la sociedad como organismo, la evolución de la socied.

Vilfredo Pareto Pareto 1848- 1923 y la ciencia el concepto de sociología

Georg Simmel 1858 1918 el pensamiento diléctico, la conciencia individual ,la sociología formal, la interacción social.

George Herbert Mead 1863 1931 la prioridad de lo social, el acto gestos y símbolos los procesos mentales, la sociedad.

Karl Mannheim 1893-1947 la sociología del conocimiento, el pensamiento conservador, idelología y utopía.

Alfred Schutz 1899 1959 interpretaciones de la obra de Schuttz, las ideas de E. Husserl, la ciencia y el mundo social.

Talcott Parsons 1902 1979 la intención integradora de Parson, los principios generales, antecedentes filosóficos y teóricos.

2. PROGRAMACIÓN IMPERATIVA

Introducción.

Es llamada así porque esta basada en comandos que actualizan variables que están en almacenamiento. Surge en 1950, cuando los programadores reconocieron que las variables y comandos de asignación constituyen una simple pero útil abstracción de memoria que se actualiza.
Los lenguajes imperativos o de procedimiento *son lenguajes controlados por mandatos u orientados a enunciados (instrucciones).* Un programa se compone de una serie de enunciados, y la ejecución de cada enunciado hace que el intérprete cambie el valor de una localidad o más en su memoria, es decir, que pase a un nuevo estado.
El desarrollo de programas consiste en construir los estados de máquina sucesivos que se necesitan para llegar a la solución. Ésta suele ser la primera imagen que se tiene de la programación, y muchos lenguajes de uso amplio (por ejemplo, C, C++, FORTRAN, ALGOL, PL/l, Pascal, Ada, Smalltalk, COBOL) manejan este modelo.

Los lenguajes de programación de la tercera generación, la cual se inició en los años 60, se conocen como lenguajes de alto nivel o de procedimientos. Una gran variedad de lenguajes se crearon para satisfacer los requisitos de las aplicaciones. Por lo que se encuentran lenguajes científicos (como el FORTRAN), lenguajes para empresas (como el COBOL) y lenguajes de uso general (incluyendo el BASIC). Los lenguajes de procedimientos deben ser traducidos al código de máquina que la computadora pueda entender. Sin embargo, estos lenguajes permiten al programador señalar cómo se debe efectuar una tarea a un nivel mayor que en los lenguajes de ensamble.
La gran mayoría de las aplicaciones de computadoras de uso en la actualidad, se escribieron utilizando lenguajes de procedimientos de la tercera generación. Se puede esperar que estos lenguajes todavía se utilicen en el futuro.

2.1. Definición

La programación imperativa se define como *un modelado de la realidad por medio de representaciones de la información y de un conjunto de acciones a realizar*. Orden de las acciones en el tiempo.

También podemos decir que es una representación simbólica de una posición de memoria que cambia de valor.

Los lenguajes imperativos son lenguajes controlados por mandatos u orientados a enunciados (instrucciones).

Su sintaxis genérica tiene la forma:

```
enunciado1;
enunciado2;
```

Ejemplos de estos lenguajes son C, C++, FORTRAN, ALGOL, PL/I, Pascal, Ada, Smalltalk, y COBOL).

Debido a su relación cerrada con las arquitecturas de las máquinas, los lenguajes de Programación Imperativa pueden ser implementados eficientemente, la razón fundamental de la Programación Imperativa esta relacionado a la naturaleza y propósito de la programación.

A. Trabajando con iteraciones: Las instrucciones que indican la repetición o iteración se llaman instrucciones iterativas. Ordenan a la UCP que itere o vuelva a ejecutar ciertas instrucciones y el número de veces respectivo. Usa variables locales para acumular él producto y para controlar las iteraciones.
B. Trabajando con recursión : No utiliza variables locales.

Paradigma imperativo

• Modelo: máquina de estados Von Neumann
• Solución del problema 'paso a paso'
• Evolución del lenguaje máquina
• Referencialmente opacos: el resultado de una expresión no es independiente del lugar donde aparece (depende de la historia; efectos colaterales).

Aspectos de Programación Imperativa (Procedimental)

1. Cualquier fragmento aislado de programa debe entenderse y mejorarse con facilidad.
2. Las dificultades comienzan cuando el efecto del cambio puede extenderse a través de un programa muy grande, tal vez introduciendo errores en "un rincón olvidado". Estos errores pueden permanecer sin detección durante años.
3. La estructuración es clave para manejar programas muy grandes. La legibilidad de un programa puede mejorarse organizándolo de tal manera que cada parte pueda entenderse en forma relativamente independiente del resto.
4. La estructura ayuda a mantener la situación dentro del límite de la atención humana.
5. Miller observó que la gente es capaz de recordar aproximadamente siete cosas, pudiendo ser bits, palabras, colores, tonos sabores, etc. Por lo tanto tentativamente podemos suponer que nuestras memorias se encuentran limitadas por el número de símbolos o unidades que podemos manejar, y no por la información que representen esos símbolos.
6. De esta manera, es beneficioso organizar de modo inteligente el material antes de tratar de memorizarlo, permitiéndonos empaquetar la misma cantidad de información en mucho menos símbolos y facilitar así la tarea de memorizar.

Características de los Lenguajes Imperativos

• Estado implícito
• Comandos o Instrucciones
  – asignación, saltos condicionales e incondicionales, bucles...
  – afectan o modifican el estado

Existen muchos lenguajes diferentes de programación imperativos u orientados a procedimientos. Algunos son ampliamente utilizados, otros se hicieron para fines específicos o, incluso, para una sola instalación u organización.

Las características generales de las instrucciones de los lenguajes de alto nivel y las características de otros lenguajes que se utilizan ampliamente, pueden agruparse en las categorías de operaciones de entrada / salida, especificaciones para el formato de datos, especificaciones aritméticas, instrucciones para la transferencia de control, especificaciones para almacenamiento, repetición e instrucciones de documentación.

Propiedades generales de los lenguajes imperativos o de procedimientos.

- Orientados a la utilización por programadores profesionales.
- Requiere especificación sobre cómo ejecutar una tarea.
- Se deben especificar todas las alternativas.
- Requiere gran número de instrucciones de procedimiento.
- El código puede ser difícil de leer, entender y mantener.
- Lenguaje creado originalmente para operación por lotes.
- Puede ser difícil de aprender.
- Difícil de depurar.
- Orientados comúnmente a archivos.

2.2. Lenguajes de programación imperativa

Se estudiarán varios lenguajes de programación utilizados comúnmente en Estados Unidos. Este análisis dará una idea acerca de cómo se usan en la elaboración de programas de cómputo para resolver problemas de procesamiento en el entorno de los negocios y la administración.

2.2.1. FORTRAN

El lenguaje de programación FORTRAN (FORmula TRANslation), diseñado en 1957, se creó originalmente para los sistemas de computación IBM. Sin embargo, su empleo se difundió rápidamente, y está disponible en casi cualquier tipo de sistema de cómputo.
Utilizado principalmente para efectuar cálculos aritméticos, algebraicos y numéricos, el FORTRAN está orientado a los procedimientos, es decir, el programa de instrucciones debe definir claramente los procedimientos aritméticos, de entrada / salida y cualquier otro

que se desee, sin preocuparse de cómo se realizan las operaciones en la unidad central de procesamiento.
En el FORTRAN, los enunciados de entrada / salida y los de FORMAT se aplican conjuntamente. Los enunciados de READ (leer) y WRITE (escribir), sirven para las funciones de entrada y salida, cada uno valiéndose de un enunciado FORMAT (formatear) que especifica el tipo de datos, su dimensión y su ubicación en los medios de entrada o de salida.
Los enunciados FORMAT son las declaraciones para especificar los datos que corresponden a las instrucciones particulares de entrada / salida.

2.2.2. BASIC

El lenguaje BASIC (acrónimo del inglés Beglnner's AII-purpose Symbolic Instruction Code) se originó en 1964 en el Dartmouth College (con el apoyo de la National Science Foundation). Este lenguaje simbólico de fácil comprensión es similar al FORTRAN. Cuando se introdujo, el BASIC fue destinado para utilizarse en los grandes sistemas de computación de tiempo compartido empleados por varios usuarios. También se le diseñó para instruir a los estudiantes sobre cómo trabajan las computadoras y cómo programarías. Así que los recursos incorporados al BASIC son muy sencillos de entender y recordar.
Desde su introducción, el BASIC ha tenido amplia aceptación en los campos de la educación, investigación e industria. Se aplica en computadoras de todos tamaños; no obstante, es el lenguaje dominante en las microcomputadoras. Muchas computadoras pequeñas incluyen el lenguaje BASIC como parte del paquete original de compra, porque es muy fácil de aprender y, también, por la poderosa capacidad de procesamiento del lenguaje en sí.
En el BASIC, a diferencia del FORTRAN no es necesario valerse de enunciados FORMAT como los de INPUT y PRINT. La computadora lee o imprime los datos directamente conforme se presentan, sin reordenarlos (editarlos) o cambiar su formato (algunas versiones del BASIC permiten el uso de las posibilidades de formato como una opción, pero no es obligatorio como en el FORTRAN).
Además de los datos numéricos, el BASIC, al igual que muchos otros lenguajes, permite el empleo de información con caracteres, llamada cadenas de caracteres. Una cadena puede ser una sola letra o símbolos (como ‘‘A’’, ‘‘B’’, ‘‘1’’, “9’’, ‘‘+‘‘ y ‘‘$‘‘), o bien diferentes caracteres utilizados en conjunto (como “ABC”, “PAGO” y “JUAN”). Adviértase que las cadenas de caracteres se encierran entre comillas y que las comas normales necesarias en la redacción en lenguaje común se colocan fuera de las comillas puesto que tales comas no forman parte de la cadena de caracteres. Así es como se manejan cadenas de datos en el BASIC. En el ejemplo de la nómina, los datos de cada empleado se tratarían como cadenas de datos.
Las operaciones aritméticas elementales en el BASIC son las mismas que en la mayoría de los otros lenguajes. Los elementos de los datos se suman, restan, dividen o multiplican, sirviéndose de los operadores bien conocidos: +, - , / y *. Se pueden incluir en una instrucción variables o constantes.
El control de la ejecución se maneja por medio de enunciados IF-THEN-ELSE. Estas instrucciones permiten al programador probar determinadas condiciones y, después, dependiendo de la condición, transferir el control a una instrucción que se encuentre en cualquier otra parte del programa. En el BASIC a todos los enunciados se les asigna

automáticamente un número de identificación, por lo que es muy fácil referirse a uno en especial.

Las condiciones se expresan en el BASIC con los siguientes operadores:
1. Mayor que: >
2. Menor que: <
3. Igual a: =
4. Mayor que o igual a: > = o =
5. Menor que o igual a: < = o =
6. Diferente de: < >

2.2.3. COBOL

El lenguaje COBOL (acróstico de COmmon Business-Oriented Language) fue creado por un comité de representantes de empresas y universidades patrocinado por el gobierno estadounidense. Las especificaciones para el lenguaje se concluyeron en 1959 y se dieron a conocer durante 1960. Continuamente mejorado y actualizado, es el lenguaje de programación dominante en el área de procesamiento de datos para las empresas en la actualidad.
Puesto que el COBOL fue diseñado explícitamente para el uso en aplicaciones de las empresas, las instrucciones están escritas en un lenguaje semejante al idioma inglés que se parece al empleado en las organizaciones. Además, el COBOL es independiente de la máquina; así, un programa escrito, por ejemplo, para una computadora IBM puede transferirse fácilmente a otros sistemas de cómputo. El COBOL es un lenguaje fácil de aprender que es eficaz si el manejo de datos alfabéticos y alfanuméricos (por ejemplo, nombres de clientes, direcciones y descripciones de mercancías) que se emplean tan a menudo en las organizaciones comerciales. Estas ventajas serán más evidentes a medida que se den detalles del programa modelo.
Los programas de COBOL se forman constructivamente. El elemento del nivel más bajo es un enunciado, que es comparable a una instrucción en el lenguaje FORTRAN. Los enunciados se agrupan en frases y las frases (sentencias) se agrupan en párrafos. Uno o más párrafos pueden utilizarse en conjunto para formar un módulo. Los siguientes niveles superiores son las secciones y las divisiones.

Los programas COBOL consisten en cuatro divisiones:

1. • División de identificación: Contiene el nombre del programa, el del programador, la fecha en que fue escrito y otra información que puede ayudar a identificar el programa y sus objetivos.
2. • División de entorno: Denomina el sistema de computo usado para compilar y ejecutar el programa; identifica los dispositivos que se utilizarán para la entrada y la salida, o bien para conjuntos de datos específicos o resultados necesarios.
3. • División de datos: Describe cuidadosamente los nombres, tipo, extensión y ubicación de los datos, y los resultados que se considerarán como entrada y salida; muestra también las relaciones entre los conjuntos de datos.
4. • División de procedimientos: contiene las frases (es decir, las instrucciones) que especifican las operaciones aritméticas y de proceso así como el orden en el que se llevarán a cabo.

El contenido de las divisiones de identificación y de entorno, se explican por sí mismas. La sección de archivo define un conjunto de datos que se introducen y los resultados que constituyen la salida (es decir, los archivos de los datos). Se utilizan membretes o etiquetas de archivo para identificar cintas y discos; ya que no se utilizan dispositivos de almacenamiento) secundario para la entrada /salida, se omiten las etiquetas de los registros de los archivos.
Los datos son ingresados y egresados en grupos llamados registros. Cada elemento de los datos que es una parte de un registro (record) (tiempo de trabajo, tasa salarial, etc.) debe darse a conocer al sistema, exactamente como se hizo en FORTRAN, por medio del enunciado FORMAT.
En COBOL esto se logra listando el nombre del elemento de datos (adviértase cómo las restricciones en la longitud de los nombres no son tan estrictas como en el FORTRAN, es posible utilizar nombres de variables que se parezcan más a las palabras reales del inglés o del español) y la descripción o especificaciones de tipo y longitud.

Un programa fuente Cobol es un conjunto de instrucciones, párrafos y secciones que se agrupan en cuatro DIVISIONES obligatorias que, escritas en orden, son las siguientes:

IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.

IDENTIFICATION DIVISION (División de identificación)

Tiene que ser incluida en cada programa fuente. Sirve para proporcionar un nombre para identificar el programa. Opcionalmente se puede especificar información acerca del autor, fecha en que fue escrito, etc.
Consta de siete posibles párrafos siendo obligatorio solamente el primero de ellos. Su formato general es el siguiente:

IDENTIFICATION DIVISION.
PROGRAM-ID. Nombre del programa.
[AUTHOR. Nombre programador.]
[INSTALLATION. Instalación.]
[DATE-WRITTEN. Fecha de escritura.]
[DATE-COMPILED. Fecha de compilación.]
[SECURITY. Comentario.]
[REMARKS. Comentario.]

PROGRAM-ID. Nombre del programa. Es el único párrafo obligatorio y sirve para especificar el nombre del programa. Este nombre será usado por el compilador o en la ejecución del programa para indicar algún error.
El nombre del programa debe ajustarse a las reglas, ya vistas, de formación de un identificador Cobol.

ENVIR0NMENT DIVISION (División de entorno)

Permite definir el tipo de ordenador para el que fue escrito el programa, así como los dispositivos periféricos necesarios para soportar los ficheros utilizados en el programa. El formato general para esta división es el siguiente:

```
ENVIRONMENT DIVISION.
[CONFIGURATION SECTION.
[SOURCE-COMPUTER. Nombre ordenador.]
[OBJECT-COMPUTER. Nombre ordenador.]
[SPECIAL-NAMES. Nombre especiales.] ]
[INPUT-OUTPUT SECTION.
FILE-CONTROL.
        Control de archivos.
[I-O CONTROL.
         Control de entrada/salida.] ]
```

CONFIGURATION SECTION (Sección de Configuración)
Esta sección es opcional. Su utilidad es indicar al programa el modelo de ordenador a utilizar y asociar nombres especiales que van a ser usados en el programa.
Para esta última opción se utiliza el párrafo SPECIAL-NAMES siendo interesante la posibilidad de intercambiar la función de la coma y el punto decimal, mediante la cláusula DECIMAL-POINT IS COMMA de la siguiente forma:

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
      DECIMAL-POINT IS COMMA.
```

INPUT-OUTPUT SECTION (Sección de entrada/salida)
Esta sección es también opcional. Proporciona información al compilador sobre los archivos utilizados en el programa y su relación con los dispositivos externos. Se explicará con detalle a la hora de trabajar con ficheros.

DATA DIVISION (División de datos)

Se utiliza para realizar una descripción completa de los ficheros que intervienen en el programa, de sus registros lógicos y de las variables de trabajo. Su formato es el siguiente:

```
DATA DIVISION.
[FILE SECTION.
[Declaración del archivo.
[Declaración del registro. ] ] …]
[WORKING-STORAGE SECTION.
[77 Declaración de variables independientes.]
[01 Declaración de registros.] ]
[LINKAGE SECTION.
[77 Declaración de variables independientes.]
[01 Declaración de registros.] ]
[COMMUNICATION SECTION.
[Descripción comunicación.]
```

[Declaración de registros.]]
[SCREEN SECTION.
[01 Descripción de pantallas.]]

FILE SECTION (Sección de Ficheros)
En esta sección se describen detalladamente toda la información referente a los archivos utilizados en el programa, así como los registros junto con sus campos y el tipo de datos que se va a almacenar en ellos. Se verá con más detalle en los apuntes de ficheros.

WORKING-STORAGE SECTION (Sección de Trabajo)
En ella se describen las variables usadas por el programa, ya sea con estructura de registro o como campos independientes.
Para declarar una variable es necesario especificar su número de nivel, su nombre, su tipo y longitud, opcionalmente también se le puede asignar un valor inicial.

2.2.4 PASCAL

Como se ha prestado cada vez más atención a la forma como se diseñan y estructuran los programas de aplicación, han surgido nuevos lenguajes que apoyan mejor el diseño estructurado. Uno de los más conocidos y más ampliamente utilizados es el llamado Pascal. Este lenguaje, que se aplica tanto en pequeñas microcomputadoras como en los sistemas de maxicomputadoras, fue introducido en 1971 por Niklaus Wirth, investigador suizo del Instituto Federal de Tecnología de Zurich. Nombró a este lenguaje en honor de Blas Pascal, el famoso matemático del siglo XVII que desarrolló la primera máquina de calcular.
Una característica importante del Pascal es la de que el programador tiene que planear el desarrollo. Por ejemplo, todas las variables y constantes se definen al principio del programa. (Otros lenguajes permiten que las variables sean introducidas en el citado programa en cualquier momento) El Pascal utiliza puntos y comas para la terminación o la separación de los enunciados (excepto para los de salida que finalizan con un punto). Los comentarios y las observaciones indican por asteriscos o se encierran entre paréntesis.
Una característica inherente de este lenguaje es su tendencia a forzar la estructuración de la lógica. Los programas constan de bloques de instrucciones que empiezan con BEGIN y acaban con END. Todo el programa se trata como si fuera un gran bloque que puede contener otros bloques envueltos o anidados.
Se ejecuta para cada registro (record) de datos en el archivo de entrada, es decir, hasta que se encuentre una marca de fin de archivo (EOF, de end-of-file) en el curso de la entrada. Todos los resultados se escriben completos con la instrucción (WRITELN), la cual contiene los nombres de las variables y los formatos de salida.
El objetivo del Pascal es semejarse en forma más precisa que otros lenguajes a la manera en que los humanos plantean los problemas. El curso lógico de las instrucciones, los diferentes bloques de funciones y de pasos y la capacidad para valerse de variables de cualquier longitud demuestran lo anterior.

Expresiones:

Las expresiones en Pascal son limitadas. No tiene expresiones condicionales de mayor tipo ni bloques de expresiones. No tiene expresiones no triviales de tipo compartido debido a que los agregados y funciones no tienen resultado de tipo compuesto.

Comandos y secuenciadores:

Comandos:

* Ejecución Condicional : { if, case }
* Ejecución Interacción
Indefinido : { repeat, while }
Definido : { for }
Estos comandos pueden ser compuestos libremente
Utilizar el secuenciador "goto". Pascal restringe el uso. Los programadores pueden usar el secuenciador "goto" de forma disciplinada

Declaraciones:

En Pascal se puede declarar :

Constantes:

```
Const
y = 20;
```

Tipos:

```
Type
Array = char [´A´...´Z´]of letras;
```

Variables :

```
Var
a,b : Integer;
c :Real;
```

Procedimientos :

```
Procedure arreglos(var x : integer);
```

Funciones :

```
Function mas(x : real) : real;
```

Pascal, soporta procedimientos y abstracción de funciones. Los procedimientos y funciones usan valores de primera clase.

2.2.5. ADA

El lenguaje más reciente orientado a utilización universal es el Ada, nombrado así en honor de la primera programadora de computación (por su trabajo con el matemático inglés Charles Babbage), Ada Augusta Byron, hija del poeta Lord Byron. La iniciativa para el desarrollo de este lenguaje nació en la Secretaría de la Defensa del gobierno de

Estados Unidos cuando intentó tener un lenguaje de programación que pudiera ser utilizado en la administración o las empresas, en la ciencia, las matemáticas y en las actividades de ingeniería. El hecho de tener un lenguaje que pudiera emplearse en muchas máquinas diversas (y, por lo tanto, haciendo a los programas de cómputo transportables de un sistema a otro) ahorraría millones de dólares cada año.
Este lenguaje, que apenas está adquiriendo una forma útil, reúne las mejores características del Pascal. También se le puede acoplar una variedad de otros componentes programáticos, según las necesidades del usuario. De esta forma, el Ada utiliza el concepto de programática "conectable" tal como los expertos en equipo de cómputo lo han tenido durante tantos años.
Se oirá hablar más acerca del Ada en los próximos años a medida que el lenguaje se utilice en muchos campos y funciones de gobierno y de la administración, desde aplicaciones empresariales hasta programas para la tecnología aeronáutica y astronáutica.

Valores, tipos y subtipos:

La elección de los tipos primitivos y compuestos es similar a Pascal.
No soporta tipos recursivos directamente, ellos tienen que ser programados usando punteros.
Soporta el concepto de subtipo, haciendo una importante distinción entre tipo y subtipo: el subtipo no es el mismo tipo.
Cada declaración de tipo crea un nuevo tipo y distinto.

Enteros -3, -2,...,2, 3, 4, 5
Subtipo {0,.., 5}
Cada valor pertenece solamente a un tipo.

Expresiones:

En ADA se puede escribir expresiones no triviales de cada tipo. En particular los agregados permiten registros y arrays para ser construidos de sus componentes. ADA no tiene expresiones condicionales o bloques de expresiones y el cuerpo de una función es un comando.

Comandos y secuenciadores:

ADA es igual a Pascal en comandos, los secuenciadores que utiliza Ada: exit y return permiten estructuras de control, además Ada hace uso del "goto".

Declaraciones:

ADA permite declarar:

-Tipos.
-Variables.
-Subtipos.
-Procedimientos.
-Constantes.
-Funciones.
*Paquetes (Packages).

*Tareas (Tasks).
*Excepciones (Exception).

Las declaraciones pueden ser localizadas en bloques de comandos, las cuales permiten que el programador restringa el ámbito de una declaración.

2.3. Selección de los lenguajes de programación

En vista de que los centros de cómputo generalmente tienen dispositivos compiladores y traductores para varios lenguajes diferentes de programación, los programadores y otros miembros del personal técnico de sistemas deben seleccionar uno de ellos para la codificación de una aplicación. (En general, los usuarios no intervienen en este proceso de selección.) Se resumirán brevemente los factores que deben considerarse en la elección de un lenguaje de programación.
La consideración fundamental (suponiendo que los programadores ya conocen los lenguajes) es la capacidad de cada lenguaje. ¿Qué tipos de necesidades de procesamiento se pueden programar con un lenguaje en particular? Después de que se conozcan las características esenciales para satisfacer la necesidad de procesamiento o manejo de datos, se deben evaluar los lenguajes en términos de:

- Especificación de entrada / salida: ¿El lenguaje respalda varios dispositivos (por ejemplo, terminal impresora, rastreador óptico, pantalla catódica)? ¿Qué tan fácil es ejecutar un volumen elevado de procesamientos con elementos de entrada / salida?
- Capacidades de manejo de datos: ¿Qué tipo de datos pueden ser procesados? ¿Cuáles son sus características de producir con nuevo formato y editado (o edición)? ¿Qué clases de conjuntos de caracteres se pueden emplear en el proceso?
- Capacidades de almacenamiento de datos: ¿Qué clases de organización de archivos (que se estudian en el siguiente capítulo) se pueden utilizar? ¿Cuál es su capacidad para almacenar y recabar registros de longitud variable? ¿Es fácil que los conjuntos de datos almacenados puedan ser creados o recabados?
- Ayudas para la programación ¿Qué tan fácil es su uso? ¿Cuáles son sus capacidades de documentación? ¿Cuáles son sus ayudas para la depuración o eliminación de los errores?
- Eficiencia: ¿Es eficiente el proceso de compilación? ¿Es eficaz el código generado durante la compilación? ¿Existe apoyo o soporte para la modificación de las características del lenguaje?
- Arquitectura del compilador
- Número de llamadas a los periféricos

Saber algo sobre las características de los lenguajes puede eliminar con rapidez algunas de ellas de la consideración para algunos tipos de procesamiento. El FORTRAN, por ejemplo, no tiene características intrínsecas para la comunicación de datos. El RPG no es adecuado para ejecutar grandes volúmenes de cálculo puro. El COBOL es conveniente para procesar grandes cantidades de datos cuando el volumen de operaciones de cálculo puro es muy bajo.

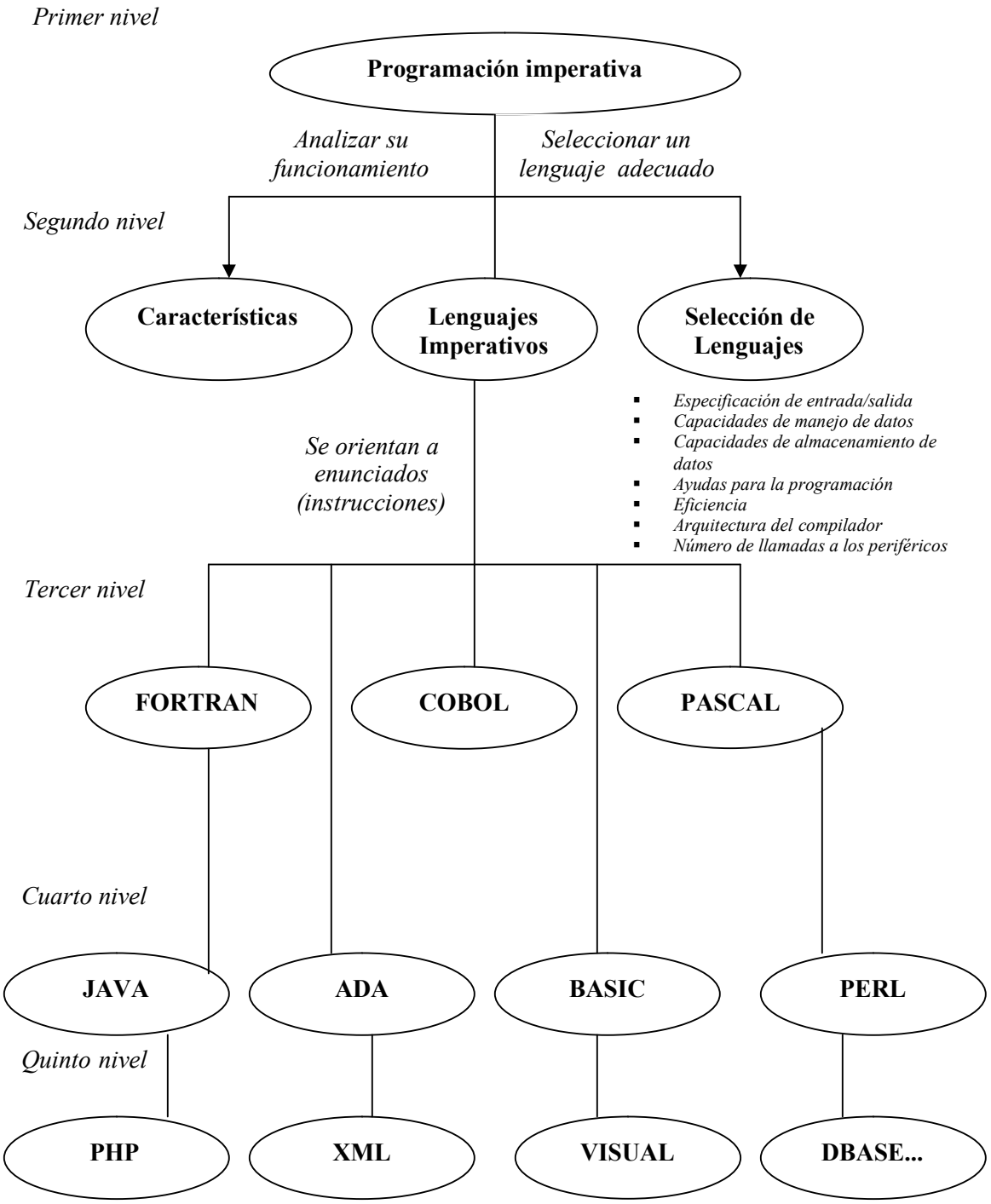
La facilidad de programación es una preocupación constante en la elección de lenguaje. Si un lenguaje es fácil de entender y aplicar, puede reducir los errores y acortar el tiempo necesario para habilitar el programa. Se tiene que considerar la facilidad de programación así como la eficiencia de la operación. Los lenguajes de alto nivel son mucho más fáciles de utilizar que el lenguaje ensamblador, pero la codificación no es tan eficiente.

El COBOL, por ejemplo, es fácil de aprender y usar, debido a que los comandos están redactados según la estructura del idioma inglés, y a la gran cantidad de instrucciones. Sin embargo, por tal volumen de instrucciones, resulta mucho menos eficaz en la ejecución que un lenguaje de ensamble. El FORTRAN es de uso mucho más eficaz (para codificar) y funciona con mayor rapidez que el COBOL, pero resulta más compleja la manipulación de datos alfabéticos.

La compatibilidad con los lenguajes que se usan en otros programas de aplicaciones puede ser también un factor decisivo en la elección. Quizá dos o más aplicaciones se combinen en un futuro próximo, por lo que se debe considerar esta posibilidad. La elección del lenguaje programático correcto es un paso muy importante en el ciclo de programación; puede facilitar la operación de la codificación para el usuario y para la organización donde se utiliza.

ACTIVIDADES COMPLEMENTARIAS

1. A partir del estudio de la bibliografía específica sugerida, elabore un mapa conceptual de los temas de la unidad.



2. Investigue cómo se utilizaría la operación de suma, resta o multiplicación, así como división y exponenciación de los números 2*3*5 en los lenguajes BASIC, COBOL, PASCAL, UNIX Y DBASE.

BASIC:

```
DEFINT A-Z
A = 2 + 3
B = 5 – 3
C = 3 / 2
D = 2 * 5
E = 2 ** 3
```

COBOL:

```
77 VAR-A  PIC 9(5) COMP.
77 VAR-B  PIC 9(5) COMP.
77 VAR-C  PIC 9(5) COMP.
77 VAR-D  PIC 9(5) COMP.
77 VAR-E  PIC 9(5) COMP.
ADD 2 TO 3 GIVING VAR-A.
SUBTRACT 3 FROM 5 GIVING VAR-B.
COMPUTE VAR-C = 3 / 2.
MULTIPLY 2 BY 5 GIVING VAR-D.
COMPUTE VAR-E = 2 ** 3.
```

PASCAL:

```
VAR A,B,C,D,E: Integer;
A:= 2 + 3;
B:= 5 - 3;
C:= 3 / 2;
D:= 2 * 5;
E:= 2 ^ 3;
```

Unix (Shell):

```
A = expr `2 + 3`
B = expr `5 – 3`
C = expr `3 \ / 2`
D = expr `2 * 5`
E = expr `2 ** 3`
```

DBASE:

```
DEFINT A-Z
```

```
A = 2 + 3
B = 5 – 3
C = 3 / 2
D = 2 * 5
E = 2 ** 3
```

3. Investigue cómo se puede diferenciar la ejecución de un comando en PASCAL y DBASE.

PASCAL:

```
System("cls");
System("del *.log");
System("otroprog.exe");
```

DBASE:

```
Control = shell("cls",1)
Control = shell("c:\appl\windows\calc.exe",1)
Control = shell("del *.log",0)
DoCmd "select nombre from empleados"
DoCmd "insert into facturas values (2.5, 8, 10.50)"
```

4. Investigue cómo se pueden aceptar datos desde el teclado en BASIC, COBOL, PASCAL y DBASE.

BASIC:

```
DIM ENTRADA$ AS STRING
PRINT "DAME UN VALOR DE ENTRADA"
INPUT$ (ENTRADA$)
```

COBOL:

```
01 ENTRADA  PIC X(50).
DISPLAY "DAME UN VALOR DE ENTRADA"
ACCEPT ENTRADA.
```

PASCAL:

```
VAR ENTRADA: CHAR;
WRITELN("DAME UN VALOR DE ENTRADA");
READLN(ENTRADA);
```

DBASE:

```
DIM ENTRADA AS STRING
```

```
PANTALLA.CAPTION = “DAME UN VALOR DE ENTRADA”)
PANTALLA.SHOW
```

Bibliografía:

- *“Lenguajes de programación”*
  Diseño e implementación
  Pratt – Zelkowitz
  Ed.Prentice may
- *“Sistemas de Información para la Administración”*
  James A. Senn
  Ed. Grupo Editorial Iberoamérica,
  México, 1990
- *“Algorítmica”*
  (Diseño y análisis de algoritmos Funcionales e Imperativos)
  Javier Calve, Juan C. González
  Ed. Addison-Wesley Iberoamericana
  México, 1993

III. Programación Orientada a Objetos.

3.1. Definición.

El Análisis Orientado a Objetos (AOO) se basa en conceptos sencillos, conocidos desde la infancia y que aplicamos continuamente: objetos y atributos, el todo y las partes, clases y miembros. La idea principal de la programación orientada a objetos es construir programas que utilizan objetos de software. Un objeto puede considerarse como una entidad independientemente de cómputo con sus propios datos y programación. Se incluirán datos (llamados campos), que describen sus atributos físicos y programación (llamados métodos), que gobierna la amnera enq ue funciona internamente y en que interactúa con otras partes relacionadas (también objetos). En la programación orientada a objetos, los objetos de software tienen una correspondiencia estrecha con los objetos reales relacionados con el área de la aplkicación. Esta correspondencia facilita la comprensión y el manejo del programa de la computadora. La programación orientada a objetos se aplica en muchos ejemplos realistas y los más grandes se ilustran con diagramas de diseño orientado a objetos.

El paradigma orientado a objetos ha sufrido una evolución similar al paradigma de programación estructurada: primero se empezaron a utilizar los lenguajes de programación estructurados, que permiten la descomposición modular de los programas; esto condujo a la adopción de técnicas de diseño estructuradas y de ahí se paso al análisis estructurado.

Son tres las características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. El software orientado a objetos permitirá que los objetos independientes se puedan ejecutar en forma simultánea, en procesadores independientes. El diseño orientado a objetos, con herramientas CASE y generadores de código, es la clave para la construcción de poderosas máquinas paralelas. La revolución industrial del software ganará fuerza cuando se difundan las técnicas orientadas a objetos y se disponga de grandes bibliotecas de clases de objetos. Las bibliotecas se enlazarán con depósitos CASE de modo que las nuevas clases se puedan ensamblar rápidamente a partir de las ya existentes.

En el Análisis Orientado a Objetos, los objetos encapsulan tanto atributos como procedimientos (operaciones que se realizan sobre los objetos), e incorpora además conceptos como el polimorfismo o la herencia que facilitan la reutilización de código. El uso de Análisis Orientado a Objetos puede facilitar mucho la creación de prototipos, y las técnicas de desarrollo evolutivo de software. Los objetos son inherentemente reutilizables, y se puede crear un catálogo de objetos que podemos usar en sucesivas aplicaciones. De esta forma, podemos obtener rápidamente un prototipo del sistema, que pueda ser evaluado por el cliente, a partir de objetos analizados, diseñados e implementados en aplicaciones anteriores. Y lo que es más importante, dada la facilidad de reutilización de estos objetos, el prototipo puede ir evolucionando hacia convertirse en el sistema final, según vamos refinando los objetos de acuerdo a un proceso de especificación incremental.

El Análisis Orientado a Objetos ofrece un enfoque nuevo para el análisis de requisitos de sistemas software. En lugar de considerar el software desde una perspectiva clásica de entrada/proceso/salida, como los métodos estructurados clásicos, se basa en modelar el sistema mediante los objetos que forman parte de él y las relaciones estáticas (herencia y composición) o dinámicas (uso) entre estos objetos. Este enfoque pretende conseguir modelos que se ajusten mejor al problema real, a partir del conocimiento del llamado ***dominio del problema***[1] , evitando que influyan en el análisis consideraciones de que estamos analizando un sistema para

[1] *El término dominio del problema o dominio de aplicación es uno de los más usados en el paradigma orientado a objetos. Se refiere al campo de aplicación del sistema, es decir, a qué es el sistema, entendido desde su propio campo de aplicación, más que a su descripción en términos de una implementación en ordenador.*

implementarlo en un ordenador. Desde este punto de vista, el Análisis Orientado a Objetos consigue una abstracción mayor que el análisis estructurado, que modela los sistemas desde un punto de vista más próximo a su implementación en una computadora (entrada/proceso/salida).

Este intento de conocer el *dominio del problema* ha sido siempre importante; no tiene sentido empezar a escribir los requisitos funcionales de un sistema de control de tráfico aéreo, y menos aún diseñarlo o programarlo sin estudiar primero qué es el tráfico aéreo o qué se espera de un sistema de control de este tipo.

El concepto renovador de la tecnología de programación orientada a objetos es la anexión de procedimientos de programas a elementos de datos. A esta nueva unión se le llama encapsulamiento y el resultado es un objeto de software. La programación orientada a objetos ha mejorado de manera esencial la creación del software, pero por sí misma no implica el mejoramiento masivo necesario para la industria de la computación. Las técnicas orientadas a objetos deben combinarse con todos los aspectos disponibles de la automatización del software.

Las técnicas orientadas a objetos modificarán:

- Toda la industria del software;
- La forma en que se venden los programas de aplicación;
- La forma de uso de las computadoras;
- La forma de uso de las redes;
- La forma de analizar los sistemas;
- La forma de diseñar sistemas;
- La forma de utilizar herramientas CASE;
- La forma de planear las empresas;
- El trabajo de todos los profesionales de los sistemas de información.

3.1.1. Ventajas del Análisis Orientado a Objetos.

- ***Dominio del problema.-*** El paradigma Orientado a Objetos es más que una forma de programar. Es una forma de pensar acerca de un problema en términos del mundo real en vez de en términos de un ordenador. El Análisis Orientado a Objetos permite analizar mejor el dominio del problema, sin pensar en términos de implementar el sistema en un ordenador. El Análisis Orientado a Objetos permite pasar directamente el dominio del problema al modelo del sistema.
- ***Comunicación.-*** El concepto Orientado a Objetos es más simple y está menos relacionado con la informática que el concepto de flujo de datos. Esto permite una mejor comunicación entre el analista y el experto en el dominio del problema (es decir, el cliente).
- ***Consistencia.-*** Los objetos encapsulan tanto atributos como operaciones. Debido a esto, el Análisis Orientado a Objetos reduce la distancia entre el punto de vista de los datos y el punto de vista del proceso, dejando menos lugar a inconsistencias o disparidades entre ambos modelos.
- ***Expresión de características comunes.-*** El paradigma Orientado a Objetos utiliza la herencia para expresar explícitamente las características comunes de una serie de objetos. Estas características comunes quedan escondidas en otros enfoques y llevan a duplicar entidades en el análisis y código en los programas. Sin embargo, el paradigma Orientado a Objetos pone especial énfasis en la reutilización, y proporciona mecanismos efectivos que permiten reutilizar aquello que es común, sin impedir por ello describir las diferencias.
- ***Resistencia al cambio.-*** Los cambios en los requisitos afectan notablemente a la funcionalidad de un sistema, por lo que afectan mucho al software desarrollado con métodos estructurados. Sin embargo, los cambios afectan en mucha menor medida a los objetos que componen o maneja el sistema, que son mucho más estables. Las modificaciones necesarias para adaptar una aplicación basada en objetos a un cambio de requisitos suelen estar mucho más localizadas.

- ***Reutilización.-*** Aparte de la reutilización interna, basada en la expresión explícita de características comunes, el paradigma Orientado a Objetos desarrolla modelos mucho más próximos al mundo real, con lo que aumentan las posibilidades de reutilización. Es probable que en futuras aplicaciones nos encontremos con objetos iguales o similares a los de la actual.

La ventaja del Análisis Orientado a Objetos es que se basa en la utilización de objetos como abstracciones del mundo real. Esto nos permite centrarnos en los aspectos significativos del dominio del problema (en las características de los objetos y las relaciones que se establecen entre ellos) y este conocimiento se convierte en la parte fundamental del análisis del sistema software, que será luego utilizado en el diseño y la implementación.

3.1.2. Ventajas de la Programación Orientada a Objetos.

- ***Simplicidad:*** como los objetos de software son modelos de objetos reales en el dominio de la aplicación, la complejidad del programa se reduce y su estructura se vuelve clara y simple.
- ***Modularidad:*** cada objeto forma una entidad separada cuyo funcionamiento interno está desacoplado de otras partes del sistema.
- ***Facilidad para hacer modificaciones:*** es sencillo hacer cambios menores en la representación de los datos o los procedimientos utilizados en un programa orientado a objetos. Las modificaciones hechas en el interior de un objeto no afectan ninguna otra parte del programa, siempre y cuando se preserve su comportamiento externo.
- ***Posibilidad de extenderlo:*** la adición de nuevas funciones o la respuesta a ambientes operativos cambiantes puede lograse con sólo introducir algunos objetos nuevos y variar algunos existentes.
- ***Flexibilidad:*** un programa orientado a objetos puede ser muy manejable al adpatrse a diferentes situaciones, pòrque es posible cambiar los patrones de interacción entre los objetos sin alterarlos.
- ***Facilidad para darle mantenimiento:*** los objetos pueden mantenerse por separado, lo que facilita la localización y el arreglo de problemas,a sí como la adición de otros elementos.
- ***Reusabilidad:*** los objetos pueden emplearse en diferentes programas. Es posible construir programas a partir de componenetes prefabricados y preprobados en una fracción del tiempo requerido para elaborar nuevos programas desde el principio.

3.1.3. Conceptos básicos.

Las técnicas orientadas a objetos se basan en organizar el software como una colección de objetos discretos que incorporan tanto estructuras de datos como comportamiento. Esto contrasta con la programación convencional, en la que las estructuras de datos y el comportamiento estaban escasamente relacionadas.

Las características principales del enfoque orientado a objetos son, en primer lugar:

- ***Identidad.-*** Los datos se organizan en entidades discretas y distinguibles llamadas objetos. Estos objetos pueden ser concretos o abstractos, pero cada objeto tiene su propia identidad. Dicho de otra forma: dos objetos son distintos incluso aún en el caso de que los valores de todos sus atributos (p. ej. nombre y tamaño) coincidan. Dos manzanas pueden ser totalmente idénticas pero no por eso pierden su identidad: nos podemos comer una u otra.
- ***Clasificación.-*** Los objetos que tengan los mismos atributos y comportamiento se agrupan en clases. Todas las manzanas tienen una serie de atributos comunes: tamaño, peso, grado de maduración, y un comportamiento común: podemos coger una manzana, moverla o comerla. Los valores de los atributos podrán ser distintos para cada una de ellas, pero todas comparten los mismos atributos y comportamiento (las operaciones que se pueden realizar

sobre ellas). Una clase es una abstracción que describe propiedades (atributos y comportamiento) relevantes para una aplicación determinada, ignorando el resto. La elección de clases es arbitraria, y depende del dominio del problema.

Según esto, una clase es una abstracción de un conjunto posiblemente infinito de objetos individuales. Cada uno de estos objetos se dice que es una instancia o ejemplar de dicha clase. Cada instancia de una clase tiene sus propios valores para sus atributos, pero comparte el nombre de estos atributos y las operaciones con el resto de instancias de su clase.

- ***Polimorfismo.-*** El polimorfismo permite que una misma operación pueda llevarse a cabo de forma diferente en clases diferentes. Por ejemplo, la operación mover, es distinta para una pieza de ajedrez que para una ficha de parchís, pero ambos objetos pueden ser movidos. Una operación es una acción o transformación que realiza o padece un objeto. La implementación específica de una operación determinada en una clase determinada se denomina método.

  Según lo dicho, una operación es una abstracción de un comportamiento similar (pero no idéntico) en diferentes clases de objetos. La semántica de la operación debe ser la misma para todas las clases. Sin embargo, cada método concreto seguirá unos pasos procedimentales específicos. Una de las ventajas del poliformismo es que se puede hacer una solicitud de una operación sin conocer el método que debe ser llamado. Estos detalles de la implantación quedan ocultos para el usuario; la responsabilidad descansa en el mecanismo de selección de la implantación orientada a objetos. La programación con objetos compatibles con conexión se logra al combinar mecanismo y técnicas de programación orientada a objetos y representa su esencia. Los ingredientes clave son:

  - *Objetos intercambiables:* antes que nada debe ser posible representar una colección de objetos similares que resulten intercambiables bajo ciertas operaciones. Tales objetos, por lo general tienen una relación es-un y pueden organizarse en jerarquías de extensión de clase en Java.
  - *Interfaces públicas uniformes:* los objetos intercambiables deben tener ciertas interfaces públicas idénticas para permitir que el mismo procedimiento polimórficofuncione en todos ellos. Esto se logra manteniendo un conjunto de métodos de interfaz pública con descriptores uniformes en toda la jerarquía de clase.
  - *Parámetros polimórficos:* un método polimórfico que opera en el objeto intercambiable, debe declarar parámetros formales capaces de recibir argumentos de cualquier tipo compatible con conexión. Los parámetros deben ser tipos de referencia a superclase.
  - *Acceso dinámico de operaciones intercambiables:* al conservar descriptores uniformes para métodos de interfaz, el mecanismo de sobreescritura de métodos de Java asegura su invocación correcta al momento de la ejecución.

- ***Herencia.-*** El concepto de herencia se refiere a la compartición de atributos y operaciones basada en una relación jerárquica entre varias clases. Una clase puede definirse de forma general y luego refinarse en sucesivas subclases. Cada clase hereda todas las propiedades (atributos y operaciones) de su superclase y añade sus propiedades particulares.

  La posibilidad de agrupar las propiedades comunes de una serie de clases en una superclase y heredar estas propiedades en cada una de las subclases es lo que permite reducir la repetición de código en el paradigma Orientado a Objetos y es una de sus principales ventajas.

3.2. Modelado.

La Técnica de Modelado de Objetos (OMT, Rumbaugh, 1991) es un procedimiento que se basa en aplicar el enfoque orientado a objetos a todo el proceso de desarrollo de un sistema

software, desde el análisis hasta la implementación. Los métodos de análisis y diseño que propone son independientes del lenguaje de programación que se emplee para la implementación. Incluso esta implementación no tiene que basarse necesariamente en un lenguaje Orientado a Objetos.

La Técnica de Modelado de Objetos es una metodología Orientado a Objetos de desarrollo de software basada en una notación gráfica para representar conceptos Orientado a Objetos. La metodología consiste en construir un modelo del dominio de aplicación y ir añadiendo detalles a este modelo durante la fase de diseño. La Técnica de Modelado de Objetos consta de las siguientes fases o etapas.

3.2.1. Fases.

- ***Conceptualización.*** Consiste en la primera aproximación al problema que se debe resolver. Se realiza una lista inicial de requisitos y se describen los casos de uso.
- ***Análisis.*** El analista construye un modelo del dominio del problema, mostrando sus propiedades más importantes. Los elementos del modelo deben ser conceptos del dominio de aplicación y no conceptos informáticos tales como estructuras de datos. Un buen modelo debe poder ser entendido y criticado por expertos en el dominio del problema que no tengan conocimientos informáticos.
- ***Diseño del sistema.*** El diseñador del sistema toma decisiones de alto nivel sobre la arquitectura del mismo. Durante esta fase el sistema se organiza en subsistemas basándose tanto en la estructura del análisis como en la arquitectura propuesta.
- ***Diseño de objetos.*** El diseñador de objetos construye un modelo de diseño basándose en el modelo de análisis, pero incorporando detalles de implementación. El diseño de objetos se centra en las estructuras de datos y algoritmos que son necesarios para implementar cada clase. OMT describe la forma en que el diseño puede ser implementado en distintos lenguajes (orientados y no orientados a objetos, bases de datos, etc.).
- ***Implementación.*** Las clases de objetos y relaciones desarrolladas durante el análisis de objetos se traducen finalmente a una implementación concreta. Durante la fase de implementación es importante tener en cuenta los principios de la ingeniería del software de forma que la correspondencia con el diseño sea directa y el sistema implementado sea flexible y extensible. No tiene sentido que utilicemos Análisis Orientado a Objetos y diseño Orientado a Objetos de forma que potenciemos la reutilización de código y la correspondencia entre el dominio del problema y el sistema informático, si luego perdemos todas estas ventajas con una implementación de mala calidad.

Algunas clases que aparecen en el sistema final no son parte del análisis sino que se introducen durante el diseño o la implementación. Este es el caso de estructuras como árboles, listas enlazadas o tablas hash, que no suelen estar presentes en el dominio de aplicación. Estas clases se añaden para permitir utilizar determinados algoritmos.

Los conceptos del paradigma Orientado a Objetos pueden aplicarse durante todo el ciclo de desarrollo del software, desde el análisis a la implementación sin cambios de notación, sólo añadiendo progresivamente detalles al modelo inicial.

3.2.2. Modelos.

Al igual que los métodos estructurados, la Técnica de Modelado de Objetos utiliza tres tipos de modelos para describir un sistema:

- ***Modelo de objetos.*** Describe la estructura estática de los objetos de un sistema y sus relaciones. El modelo de objetos contiene diagramas de objetos. Este modelo muestra la estructura estática de los datos del mundo real y las relaciones entre estos datos. El modelo de objetos precede normalmente al dinámico y al funcional porque normalmente está mejor definido en la especificación preliminar, es menos dependiente de detalles de la

aplicación, es más estable respecto a la evolución de la solución y es más fácil de entender que el resto. Un diagrama de objetos es un grafo cuyos nodos son clases y cuyos arcos son relaciones entre clases.

- ***Modelo dinámico.*** El modelo dinámico describe las características de un sistema que cambia a lo largo del tiempo. Se utiliza para especificar los aspectos de control de un sistema. Para representarlo utilizaremos DEs.
- ***Modelo funcional.*** Describe las transformaciones de datos del sistema. El modelo funcional contiene DFDs y especificaciones de proceso.

Los tres modelos son vistas ortogonales (independientes) del mismo sistema, aunque existen relaciones entre ellos. Cada modelo contiene referencias a elementos de los otros dos. Por ejemplo, las operaciones que se asocian a los objetos del modelo de objetos figuran también, de forma más detallada en el modelo funcional. El más importante de los tres es el modelo de objetos, porque es necesario describir qué cambia antes que decir cuándo o cómo cambia.

Modelo de objetos.

Los cambios y las transformaciones no tienen sentido a menos que haya algo que cambiar o transformar. La Técnica de Modelado de Objetos considera este modelo el más importante de los tres. El modelo de objetos se representa gráficamente con diagramas de objetos y diagramas de instancias, que contienen clases de objetos e instancias, respectivamente. Las clases se disponen en jerarquías que comparten una estructura de datos y un comportamiento comunes, y se relacionan con otras clases. Cada clase define los atributos que contiene cada uno de los objetos o instancias y las operaciones que realizan o sufren estos objetos.

El modelo de objetos puede contener los siguientes elementos:

Instancias.		Cada uno de los objetos individuales.
Clases.		Abstracción de objetos con propiedades comunes.
	Atributos.	Datos que caracterizan las instancias de una clase.
	Operaciones.	Funciones que pueden realizar las instancias
Relaciones.		Se establecen entre clases.
Asociación.		Relación de uso en general.
	Multiplicidad.	Número de instancias que intervienen en la relación.
	Atributos.	Algunos atributos pueden depender de la asociación.
	Calificación.	Limita la multiplicidad de las asociaciones
	Roles.	Indican los papeles de las clases en las relaciones
	Restricciones y ordenación	
Composición.		Relaciones todo/parte.
Generalización.		Relaciones padre/hijo
	Redefinición.	Modificación de las propiedades heredadas.
Enlaces.		Instancias de una relación. Relaciona instancias.

Los objetos del modelo pueden ser tanto entidades físicas (como personas, casas y máquinas) como conceptos (como órdenes de compra, reservas de asientos o trayectorias). En cualquier caso, todas las clases deben ser significativas en el dominio de aplicación. Hay que evitar definir clases que se refieren a necesidades de implementación como listas, subrutinas o

el reloj del sistema. No todas las clases figuran explícitamente en la especificación preliminar, algunas están implícitas en el dominio de aplicación o son de conocimiento general.

Podemos empezar haciendo una lista de clases candidatas a partir de la especificación preliminar. Normalmente las clases se corresponden con nombres en este documento. No hay que preocuparse aún de establecer relaciones de generalización/especialización entre las clases. Esto se hace para reutilizar código y estas relaciones aparecen más claramente cuando se definan atributos y operaciones.

Una vez que tenemos una lista de clases candidatas hay que proceder a revisarla, eliminando las incorrectas, siguiendo los siguientes criterios:

- ***Clases redundantes.*** Si dos clases representan la misma información nos quedaremos con la que tenga un nombre más significativa, eliminando la otra. (p. ej. Cliente y Usuario).
- ***Clases irrelevantes.*** Podemos haber incluido clases que sean importantes en el dominio de aplicación, pero que sean irrelevantes en el sistema que pretendemos implementar. Esto depende mucho del problema concreto.
- ***Clases demasiado generales.*** Las clases deben ser específicas. Deben tener unos límites claros y unos atributos y un comportamiento comunes para todas las instancias. Debemos eliminar las clases demasiado generales, normalmente creando otras más específicas. Las clases genéricas se incluirán más adelante si es necesario, al observar atributos u operaciones comunes a varias clases.
- ***Atributos.*** Los nombres que describen propiedades de los objetos son atributos, no clases. Una de las características de un objeto es su identidad (aún cuando los valores de los atributos sean comunes), y otra muy común es su existencia independiente. Los atributos de un objeto no presentan estas dos propiedades. Si la existencia independiente de un atributo es importante, hay que modelarlo como una clase. (P. ej. podemos considerar el Despacho, como un atributo de la clase Empleado, pero esto nos impide hacer una reasignación de despachos, por lo que, si queremos implementar esta operación, deberemos considerar los despachos como objetos en lugar de como atributos).
- ***Operaciones.*** Una clase no puede ser una operación que se aplique sobre los objetos sin tener independencia por sí misma (p. ej. en nuestro modelo de la línea telefónica la Llamada es una operación, no una clase). Sin embargo, esto depende del dominio de aplicación: en una aplicación de facturación telefónica, la Llamada será una clase que contiene atributos tales como Origen, Destino, Fecha, Hora y Número de Pasos.
- ***Roles.*** El nombre de una clase debe depender de su naturaleza intrínseca y no del papel que juegue en el sistema. Podemos agrupar varias clases en una, si intrínsecamente son lo mismo, y luego distinguir estos roles en las asociaciones. (P. ej. La clase Persona puede jugar varios papeles (Propietario, Conductor, etc.) en relación con la clase Vehículo. En otros casos, una misma entidad física puede modelarse mediante varias clases distintas si los atributos y el comportamiento (y especialmente las instancias) son distintos dependiendo del papel que juegue en el sistema.
- ***Objetos internos.*** No debemos incluir en el la fase de análisis clases que no se corresponden con el dominio de aplicación sino que tienen relación con la implementación del sistema.

Una vez identificadas las clases debemos empezar a preparar el diccionario de datos del sistema. Cada clase figurará como una entrada en el diccionario donde se define brevemente su significado y las funciones que realiza. El diccionario de datos también contiene entradas para las asociaciones, las operaciones y los atributos, que deben ser definidos según las vamos incorporando al modelo.

3.3. Relaciones entre objetos (Estáticas y Dinámicas)

El Análisis Orientado a Objetos ofrece un enfoque nuevo para el análisis de requisitos de sistemas software. En lugar de considerar el software desde una perspectiva clásica de entrada/proceso/salida, como los métodos estructurados clásicos, se basa en modelar el sistema mediante los objetos que forman parte de él y las relaciones estáticas (herencia y composición) o dinámicas (uso) entre estos objetos. Este enfoque pretende conseguir modelos que se ajusten mejor al problema real, a partir del conocimiento del llamado dominio del problema, evitando que influyan en el análisis consideraciones de que estamos analizando un sistema para implementarlo en un ordenador. Desde este punto de vista, Análisis Orientado a Objetos consigue una abstracción mayor que el análisis estructurado, que modela los sistemas desde un punto de vista más próximo a su implementación en una computadora (entrada/proceso/salida).

Este intento de conocer el dominio del problema ha sido siempre importante; no tiene sentido empezar a escribir los requisitos funcionales de un sistema de control de tráfico aéreo, y menos aún diseñarlo o programarlo sin estudiar primero qué es el tráfico aéreo o qué se espera de un sistema de control de este tipo.

En programamción, la posibilidad de determinar la clase de un objeto en tiempo de ejecución, y de asignarle almacenamiento, se conoce como ***ligadura dinámica***. La ligadura dinámica también se conoce como *ligadura posterior* y es la técnica de programación que se emplea usualmente para realizar el poliformismo en los lenguajes de programación orientado a objetos.. En los lenguajes limitados estáticamente el compilador asigna el almacenamiento a los objetos, y su tipo únicamente determina su clase. Una clase tiene miembros y operaciones, mientras que un conjunto sólo tiene miembros. La ***ligadura previa*** o ***estática,*** es en la que el compilador lleva a cabo la asignación de tipos.

3.4. Clases, Instancias y Objetos.

Clases.- Una clase o clase de objetos es una abstracción que describe un grupo de instancias con propiedades (atributos) comunes, comportamiento (operaciones) común, relaciones comunes con otros objetos y (lo que es más importante) una semántica común. Así un caballo y un establo tienen los dos un coste y una edad, pero posiblemente pertenezcan a clases diferentes (aunque esto depende del dominio de aplicación: en una aplicación financiera ambos pertenecerían posiblemente a la misma clase: Inversiones). Todas las clases deben ser significativas en el dominio de aplicación. Hay que evitar definir clases que se refieren a necesidades de implementación como listas, subrutinas o el reloj del sistema. No todas las clases figuran explícitamente en la especificación preliminar, algunas están implícitas en el dominio de aplicación o son de conocimiento general. El símbolo gráfico para representar clases es un rectángulo, en el que figura el nombre de la clase. Las clases se representan en los diagramas de clases, que son plantillas que describen un conjunto de posibles diagramas de instancias. Describen, por tanto el caso general.

Podemos empezar haciendo una lista de clases candidatas a partir de la especificación preliminar. Normalmente las clases se corresponden con nombres en este documento. No hay que preocuparse aún de establecer relaciones de generalización/especialización entre las clases. Esto se hace para reutilizar código y estas relaciones aparecen más claramente cuando se definan atributos y operaciones.

La mayoría de las ***instancias***[2] de una clase derivan su individualidad de tener valores diferentes en alguno/s de sus atributos o de tener relaciones con instancias diferentes. No

[2] *instancia*, a pesar de ser el término utilizado habitualmente en la bibliografía en castellano, es una mala traducción del término inglés *instance*, siendo *ejemplar* la traducción correcta. Algunos autores solventan el problema hablando de clases y objetos, pero esto último resulta a veces ambiguo.

obstante pueden existir instancias con los mismos valores de los atributos e idénticas relaciones. El símbolo gráfico para representar instancias es un rectángulo de esquinas redondeadas. Dentro del rectángulo figura la clase a la que pertenece la instancia (entre paréntesis) y los valores de sus atributos.

Las instancias figuran en diagramas de instancias, que se utilizan normalmente para describir ejemplos que aclaren un diagrama de objetos complejos o para describir escenarios determinados (p. ej. situaciones típicas o anómalas, escenarios de prueba, etc.). La diferencia entre instancia y clase está en el grado de abstracción. Un objeto es una abstracción de un objeto del mundo real, pero una clase es una abstracción de un grupo de objetos del mundo real. La abstracción permite la generalización y evita la redefinición de las características (atributos, comportamiento o relaciones) comunes, de forma que se produce una reutilización de estas definiciones comunes por parte de cada uno de los objetos. Por ejemplo todas las elipses (instancias) comparten las mismos operaciones para dibujarlas o calcular su área.

Objetos.- Los objetos del modelo pueden ser tanto entidades físicas (como personas, casas y máquinas) como conceptos (como órdenes de compra, reservas de asientos o trayectorias). En cualquier caso, todas las clases deben ser significativas en el dominio de aplicación.Durante el proceso de desarrollo aparecen tres categorías de objetos:

- ***Los objetos del dominio*** son significativos desde el punto de vista del dominio del problema. Existen de forma independiente a la aplicación y tienen sentido para los expertos del dominio.
- ***Los objetos de aplicación*** representan aspectos computacionales de la aplicación que son visibles para los usuarios. No existen en el espacio del problema, solo tienen sentido en el contexto de la aplicación que vamos a desarrollar para solucionarlo. Sin embargo, no dependen exclusivamente de decisiones de diseño, puesto que son visibles al usuario y no pueden ser cambiados sin alterar la especificación de la aplicación. No pueden obtenerse analizando el dominio de la aplicación, pero sí pueden ser reutilizados de aplicaciones anteriores, incluso aunque sean de diferente dominio. Los objetos de aplicación incluyen controladores, dispositivos e interfaces.
- ***Los objetos internos*** son componentes de la aplicación que resultan invisibles para el usuario. Su existencia se deriva de decisiones de diseño para implementar el sistema. No deben aparecer durante el análisis. Una parte importante del diseño consiste en añadir objetos internos para hacer factible la implementación del sistema.

Por tanto, en el modelo de objetos figurarán tanto objetos del dominio como objetos de aplicación. Su construcción puede ser realizada en dos fases:

- En una primera fase podemos construir un modelo que represente los objetos del dominio del problema y las relaciones que existen entre ellos. Este modelo equivale a lo que se llama modelo esencial en la terminología del análisis estructurado.
- En una segunda fase podemos construir un modelo de aplicación, completando el modelo anterior con objetos de aplicación. Para esto jugarán un papel muy importante los casos de uso desarrollados durante la conceptualización del problema.

3.5. Interfaz.

Los métodos abstractos son útiles cuando se quiere que cada implementación de la clase parezca y funcione igual, pero necesita que se cree una nueva clase para utilizar los métodos abstractos. Los interfaces proporcionan un mecanismo para abstraer los métodos a un nivel superior. Un interface contiene una colección de métodos que se implementan en otro lugar.

La razón por la que la aplicación más popular de la programación orientada a objetos hasta los últimos tiempos ha sido el diseño de interfaces de usuario (IU) es, en parte, la amplia influencia de Smalltalk, que hacía hincapié en la interfaz de usario y en las aplicaciones de automatización de oficinas desde los primeros días de Dynabook, y en parte, también, como consecuencia de la complejidad de las interfaces gráficas de usuario típicas (GUI).

Los beneficios de la programación orientada a objetos que son aplicables específicamente en la zona de los Interfaces de Usuario son como siguen:

- La facilidad con la cual es posible utilizar y reutilizar una interfaz congruente entre objetos de “escritorio”.
- El hecho consistente en que las interfaces poseen relativamente pocas clases.
- El hecho de que la metáfora del escritorio se corresponde, de forma relativamente sencilla, con la metáfora de los objetos.
- Como consecuencia del establecimiento de la tecnología en este aspecto, existe un suministro abundante de bibliotecas útiles de objetos, y Smalltalk y otros lenguajes incluyen, en la actualidad, algunas de estas características como estándares.
- La mera complejidad de lo GUI modernos hace que la programación orientada a objetos sea una necesidad más que un lujo: si no existiese, entonces, los GUI se verían obligados a inventarla.

3.6. Clase, Estructuras de datos abstractos y Tipos de datos abstractos.

En el análisis orientado a objetos, la estructura es un conjunto enlazado de objetos. Las estructuras son de tres tipos principales:

- De clasificación.- Dícese de una estructura en forma de árbol o de red que está basada eb las primitivas semánticas de inclusión (una clase de) y de pertenencia (IsA) y que indica la herencia puede rrealizar la especificación o la realización. Los objetos pueden participar en más de una de estas estructuras, dando lugar a herencia múltiple.
- De composición.-
- De uso.- Dícese de la estructura de relaciones ecxistente entre clientes y servidores que están conectados mediante el paso de mensajes.

Las asociaciones generales también dan lugar a estructuras.

Un tipo de datos abstractos (abstract data type, ADT) es una abstracción, similar a una clase, que describe un conjunto de objetos en términos de una estructura de datos encapsulada u oculta y las operaciones sobre esa estructura. Los tipos de datos abstractos pueden ser definidos por el usuario al construir una aplicación, en lugar de ser construidos por el diseñador del lenguaje subyacente. Los tipos de datos abstractos incluyen métodos. Por ejemplo un tipo de datos abstractos que represente longitudes expresadas en unidades inglesas incluría métodos para sumar pies y pulgadas.

Los tipos de datos abstractos (abstract data type, ADT) extienden el concepto de los tipos definidos por el usuario (TDU), añadiendo el encapsulado. Los tipos de datos abstractos contienen la representación y operaciones de un tipo de datos, sino que proporciona un muro de protección que impide el uso inadecuado de sus objetos. Toda la interfaz ocurre a través de operaciones definidas dentro del tipo de datos abstractos. Así las operaciones proporcionan un medio bien definido para tener acceso a los objetos de un tipo de datos. Los tipos de datos abstractos san a los objetos una interfaz pública a través de sus operaciones permitidas. Sin embargo, las representaciones y el código ejecutable (el método) de cada operación son privadas.

La capacidad de uso de los tipos de datos abstractos apareció por primera vez con Simula 67. Su implantación se llama *class* (clase). Modula se refiere a su implantación de los tipos de datos abstractos como un *module* (módulo), en tanto que ADA utiliza la palabra *package* (paquete). En todos los casos, los tipos de datos abstractos proporcionan una forma para que los diseñadores de sistemas identifiquen los tipos de datos del mundo real y los empaquen en forma más conveniente y compacta.

Los tipos de datos abstractos se pueden definir para cosas tales como fecha, paneles de la pantalla, pedidos de clientes y solicitudes de partes. Una vez definidos, el diseñador se puede dirigir a los os tipos de datos abstractos en operaciones futuras. El tipo de datos abstractos se define también mediante un conjunto de operaciones permisibles. Estas operaciones proporcionan una especie de armadura que protege a la estructura esencial del uso arbitrario y accidental. Además, cada método o algoritmo de procesamiento utilizado para llevar a cabo una operación queda oculto a los usuarios. Lo que el usuario debe proporcionar es un objeto apropiado para llamar, o solicitar, la operación junto con los parámetros permitidos y aplicables.

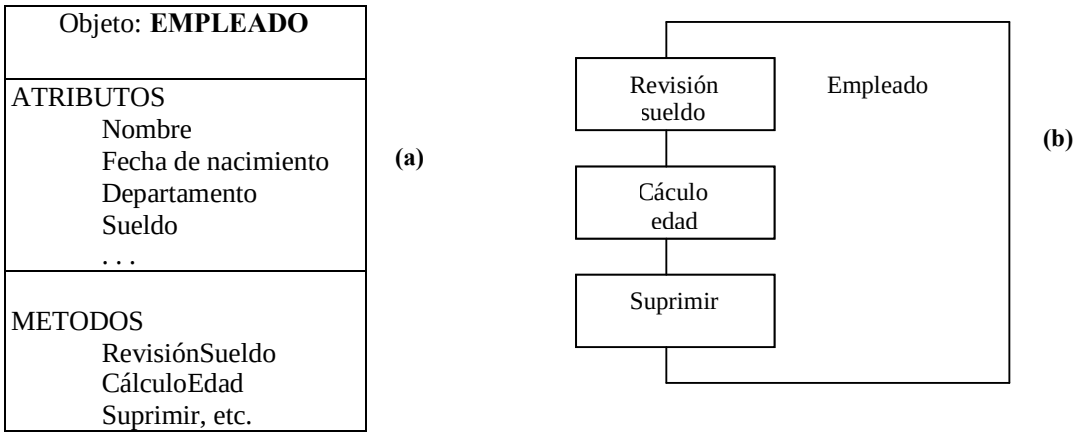


Figura 3.6.1. **(a)** Objeto o tipo de datos abstracto que encierra el concepto de empleado. **(b)** La clase Empleado, en la notación de orientación a objetos más común.

3.7. Clases Concretas, Abstractas, Metaclases, Superclases, Subclases.

Una clase abstracta es una clase que no tiene instancias directas pero cuyas clases descendientes (***clases concretas***) sí pueden tener instancias directas. Una clase concreta puede tener subclases abstractas, pero éstas han de tener subclases concretas (las hojas de una jerarquía de clases han de ser clases concretas).

Las clases abstractas organizan características comunes a varias clases. A veces es útil crear una superclase abstracta que encapsule aquellas características (atributos, operaciones y asociaciones) comunes a un conjunto de clases. Puede ser que esta clase abstracta aparezca en el dominio del problema o bien que se introduzca artificialmente para facilitar la reutilización de código.

Normalmente las clases abstractas se usan para definir métodos que son heredados por sus subclases. Sin embargo, una clase abstracta puede definir el protocolo de una determinada operación sin proporcionar el correspondiente método. Esto de denomina operación abstracta, que define la forma de una operación para la que cada subclase concreta debe suministrar su propia implementación. Una clase concreta no puede tener operaciones abstractas porque los objetos de esta clase tendrían operaciones indefinidas. Para indicar que una operación es abstracta se coloca una restricción {abstracta} junto a ella.

Una de las características más útiles de cualquier lenguaje orientado a objetos es la posibilidad de declarar clases que definen como se utiliza solamente, sin tener que implementar métodos. Esto es muy útil cuando la implementación es específica para cada usuario, pero todos los usuarios tienen que utilizar los mismos métodos. Cuando una clase contiene un método abstracto tiene que declararse abstracta. No obstante, no todos los métodos de una clase abstracta tienen que ser abstractos. Las clases abstractas no pueden tener métodos privados (no se podrían implementar) ni tampoco estáticos. Una clase abstracta tiene que derivarse obligatoriamente, no se puede hacer un new de una clase abstracta.

En lenguajes orientados a objetos (como el SmallTalk), las ***metaclases*** (*metaclass*) son aquellas clases cuyas instancias son clases, por oposición a una clase abastracta, que no posee instancias, sino cólo subclases. Se utiliza de modo impreciso como sinónimo de clase abstracta.

Una ***superclase*** es una clase con uno o más miembros que son clases más especializadas a su vez.. Las ***subclases*** (subclass) se consideran de una clase que posee un enlace AKO[3] con una clase más general, tal como en "un PERRO es una clase de MAMÍFERO".

3.8. Generalización, Especialización Redifinición de clases.

La ***generalización*** es el resultado (o el acto) de distinguir un tipo de objeto como más general, o incluso, que es más que otro. Todo lo que se aplique a un tipo de objeto también se aplica a sus subtipos. Cada instancia de un tipo de objeto es también una instancia de sus supertipos.

Las jerarquías de generalización son importantes para el desarrollador orientado a objetos por dos razones. La primera es que el uso de supertipos y subtipos proporciona una herramienta útil para describir el mundo del sistema de aplicación. La seguridad es que indica las direcciones de herencia entre las clases en los lenguajes de programación orientada a objetos.

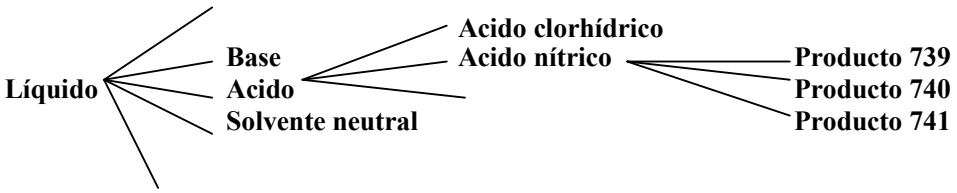
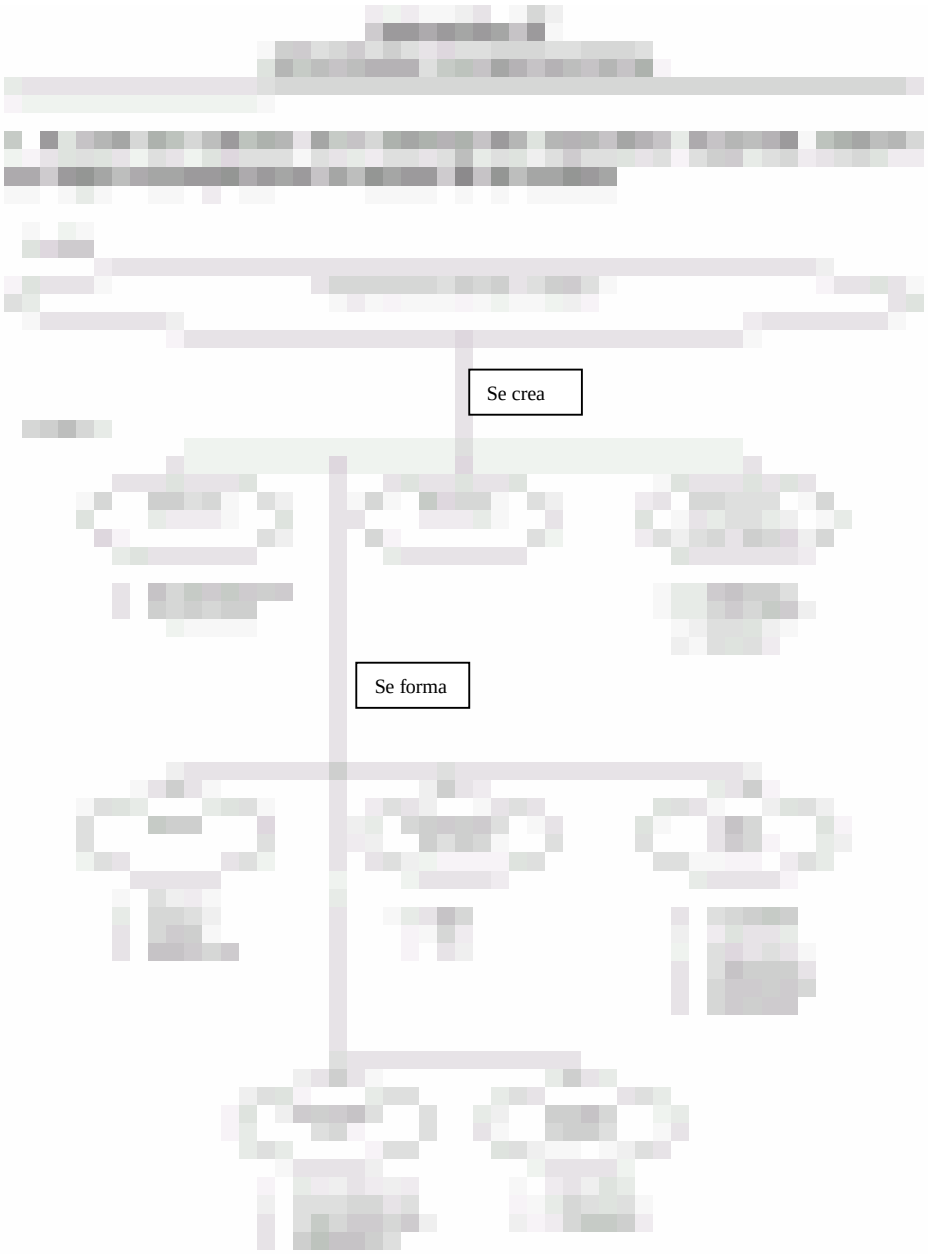
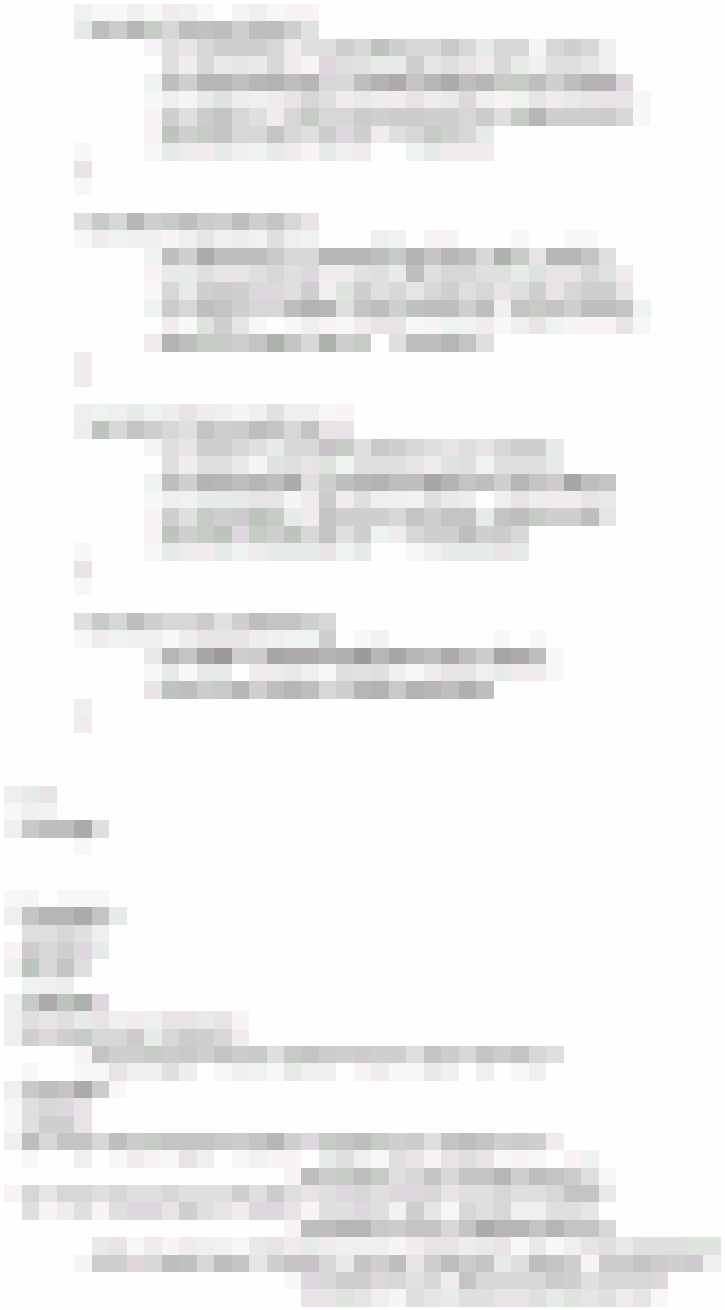
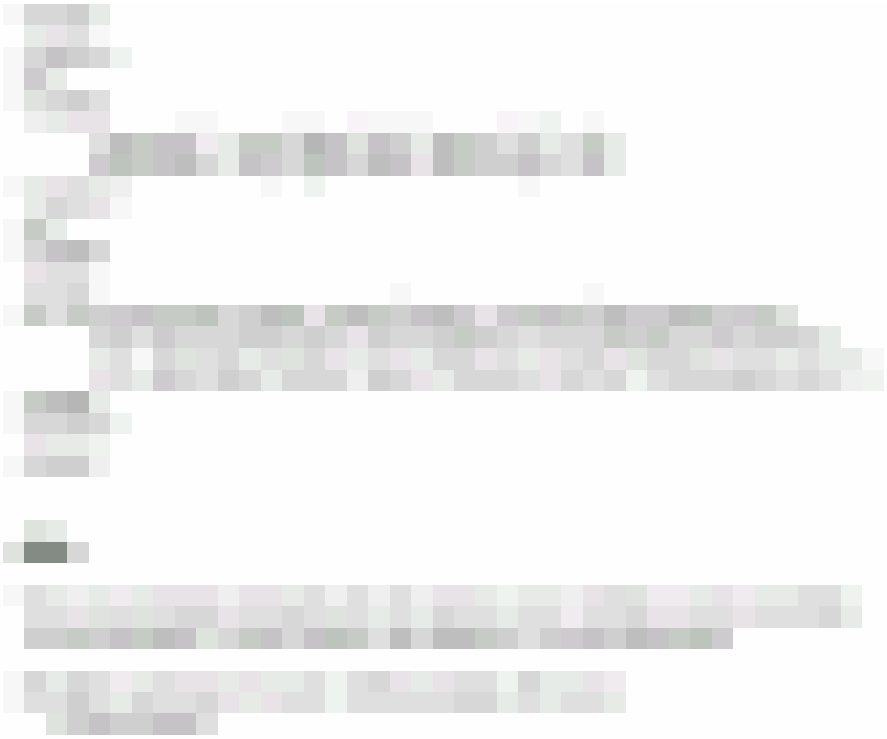


Fig. 3.8.1. Una ***Jerarquía de generalización*** que indica que todas las propiedades de ácido nítrico se aplican a **producto 739, producto 740 y producto 741**, pero cada producto tiene sus propiedades particulares. Del mismo modo, todas las propiedades de ácido se aplican a ***ácido nítrico, ácido clorhídrico***, etcétera.

[3] AKO (a kind of) Un-tipo-de se refiere a la relación de herencia que media entre las clases y sus superclases.


Se crea
Se forma





BIBLIOGRAFÍA

- **JAVA, CON PROGRAMACION ORIENTADA A OBJETOS Y APLICACIONES EN LA WWW**
  WANG, PAUL S.
  EDITORIAL INTERNATIONAL THOMSON EDITORES
  PRIMERA EDICION
  MEXICO, 1999

- **ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS**
  MARTIN, JAMES
  ODELL, JAMES J.
  EDITORIAL PRENTICE HALL
  PRIMERA EDICION
  MEXICO, 1994

- **METODOS ORIENTADO A OBJETOS**
  GRAHAM, IAN
  EDITORIAL ADDISON-WESLEY/DIAZ DE SANTOS
  PRIMERA EDICION
  MEXICO, 1996

- **WWW. MONOGRAFIAS.COM**
  APUNTES DE INFORMATICA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

DIVISION DEL SISTEMA DE UNIVERSIDAD ABIERTA

INFORMATICA IV

UNIDAD 4: PROGRAMACIÓN FUNCIONAL

PROFESOR:

HERNÁNDEZ CASTILLO VICENTE

Informática IV

4. Programación Funcional

Definición

La programación funcional se basa en el uso de las funciones matemáticas para producir mejores efectos y que sus resultados sean más eficientes. La función se define basándose en procedimientos y ésta puede ser recursiva.

Puede ser predefinida por el lenguaje, como en Pascal: la raíz cuadrada de x (square X) o programada en forma funcional, donde los valores son producidos por una función y estos se pasan por unos parámetros a otra función.

Una función es una regla para mapear (o asociar) los miembros de un grupo (grupo dominante) a aquellos de otro (grupo de rango). Una *definición de función,* especifica el dominio, el rango y la regla de mapeo para la función. Una vez que la función ha sido definida, puede ser aplicada a un elemento particular del grupo de dominio: produce la aplicación (o resultados, o regreso) al elemento asociado en el grupo de rango.

Un lenguaje funcional tienen cuatro componentes:

1. Un grupo de funciones primitivas.- Son predefinidas por el lenguaje y pueden ser aplicadas.

2. Un grupo de formas funcionales.- Son mecanismos a través de los cuales las funciones pueden ser combinadas para crear nuevas funciones.

3. La operación de la aplicación.- Es la construcción de un mecanismo para aplicar una función a sus argumentos y producir un valor.

4. Un grupo de datos objetos.- Son los miembros que permiten los grupos de dominio y rango. La característica de los lenguajes funcionales es proveer un grupo muy limitado de datos objetos con una simple y regular estructura. La esencia de la programación funcional es definir nuevas funciones usando las formas funcionales.

En general, la sintaxis de esta clase de lenguajes es similar a:

función (.. función 2(función1(datos))...)

LISP y ML son dos lenguajes funcionales que manejan este modelo. Ambos difieren de otros lenguajes anteriores en dos aspectos importantes:

- Desde el punto de vista de su diseño, están destinados a programarse en forma aplicativa. La ejecución de una expresión es el concepto principal que se necesita para la secuenciación de programas, en vez de los enunciados de lenguajes como C, Pascal o Ada.

- Puesto que son aplicativos (o funcionales), el almacenamiento en montículos y las estructuras de lista se convierten en el proceso natural para gestión de almacenamiento en lugar del mecanismo tradicional de registros de activación de los lenguajes de procedimiento.

Características:

- El valor de una expresión depende sólo de los valores de sus sub-expresiones si las tiene.

- El manejo de almacenamiento es implícito. Ciertas operaciones integradas de los datos asignan almacenamiento en el momento necesario. El almacenamiento que se vuelve inaccesible se libera, en forma automática.

- La programación funcional trata a las funciones como "ciudadanos de primera clase". Las funciones son valores de primera clase. Las funciones tienen la misma jerarquía que cualquier otro valor. Una función puede ser el valor de una expresión, puede pasarse como argumento y puede colocarse en una estructura de datos.

Paradigma funcional

• Aplicación y composición de funciones.

• Ejecutar un programa: **evaluar** una función.

• El orden de evaluación de las funciones **no** depende del programador (con reservas).

• Diferencia clara entre datos de entrada y de salida (una única dirección).

Principales ventajas:

1. Elegancia, claridad, sencillez, potencia y concisión
2. Semánticas claras, simples y matemáticamente bien fundadas
3. Cercanos al nivel de abstracción de las especificaciones formales / informales de los problemas a resolver
4. Referencialmente transparentes: Comportamiento matemático adecuado que permite razonar sobre los programas
5. Soportan técnicas muy avanzadas de desarrollo, mantenimiento y validación de programas
6. Altas dosis de paralelismo implícito
7. Aplicaciones variadas y de gran interés
8. Provee de un paradigma para programar en paralelo.
9. Es ampliamente aplicada en la Inteligencia Artificial.
10. Es bastante útil para el desarrollo de prototipos.

Aplicaciones:

- Interfaces para bases de datos (SQL es “casi” declarativo)
- Ingeniería de lenguaje natural y consulta de bases de datos en lenguaje natural
- Sistemas de planificación y scheduling
- Sistemas de tomas de decisiones basados en el conocimiento
- Sistemas de computación simbólica (Mapple o Mathematica son “casi” declarativos)
- Aplicaciones de aprendizaje y enseñanza
- Descripción de lenguajes y construcción de compiladores
- Verificación de propiedades (demostración de teoremas, verificación de sistemas hardware/software, diagnosis, etc.)

- Enrutado de aviones para Alitalia (Prolog), para Airbone Corps USA (Prolog), asignación de aviones a puertas de embarque - APACHE (CHIP)
- Diseño de circuitos lógicos para protección automática de trenes en la estación de Bolonia.
- Configuración de teléfonos móviles - Nokia (IF-Prolog)
- Configuración de centrales telefónicas – Ericsson (Erlang)
- Acceso a bases de datos en español - Software AG España (Haskell)

4.2. Estructura de los lenguajes de programación funcional

Expresiones y sentencias.

Expresiones:

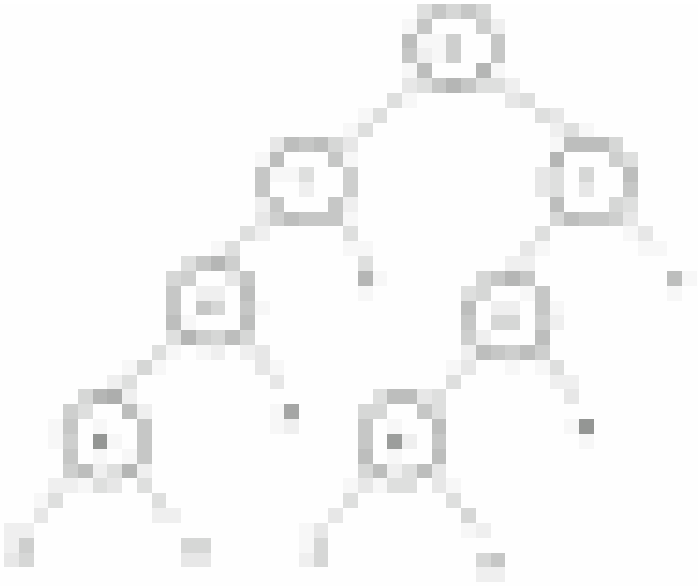
(a+b) x c	aritmética.
(a+b) = 0	relacional.
~(a V b)	lógica

El propósito es asignar un valor a través del proceso de evaluación.

Los lenguajes de programación incluyen también sentencias: unas para alterar el flujo del programa y otras para alterar el estado de la computadora.

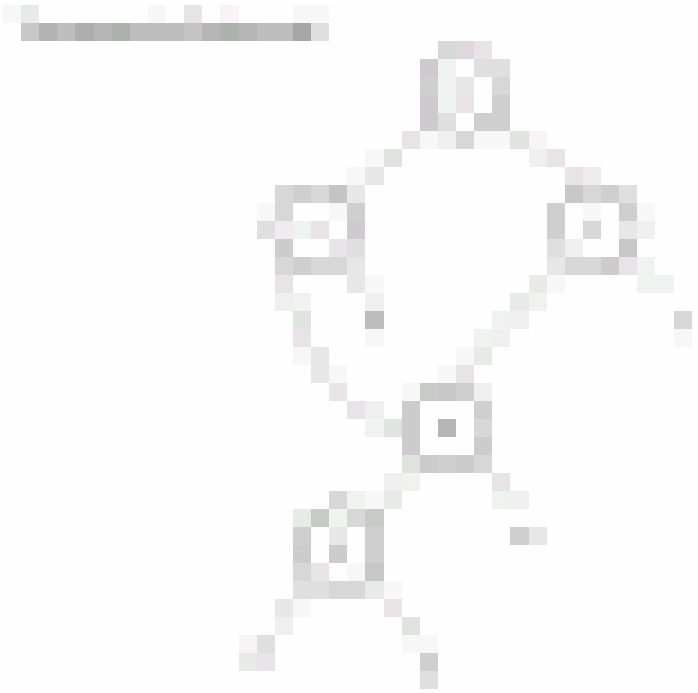
```
If x > 0 then
S := 1
Else
S:= -1
End if;
I := I +1;
```

Independencia en el orden de evaluación.
(2 a x + b)(2 a x + c) = 0



La independencia de evaluación es también llamada Propiedad Church – Rosser.
Funciones puras.

```
Int f(int x)
{
return (x + x );
}
```



Transparencia Referencial.

Interfaces Manifiestas:

Son las conexiones de entrada /salida entre subexpresiones y sus expresiones allegadas, obviamente son visuales.

Principios de Estructuración de Hoare.

1. Transparencia de Significado.
2. Transparencia de propósito.
3. Independencia de partes.
4. Aplicación recursiva.
5. Interfaces Estrechas.
6. Manifestación de estructuras.

Definición de funciones.

Una función es sólo un conjunto de parejas de entrada-salida.

Estructura del lenguaje:

- Tipos de datos (particularmente secuencias).
- Operaciones
- Todos los lenguajes de programacion comprenden dos partes:
- Uno independiente del dominio de la aplicación intentada.
- Otro dependiente sobre el dominio de la aplicación.

- Parte independiente del dominio(framework).-- (Proveen la forma del lenguaje) Incluye los mecanismos básicos de lingüistica utilizados para la construcción de programas.

Para Lenguajes imperativos:

- Estructuras de control
- Mecanismos de procedimientos
- Operaciones de asignación

Para Lenguajes aplicativos:

- Definición de funciones
- Mecanismos de aplicación de funciones.
- Otras construcciones como condicionales pueden incluirse en las definiciones o en las aplicaciones de las funciones.
- Parte dependiente del dominio. -- (Proveen el contenido del lenguaje)
- Es un conjunto de componentes que son completamente usados en el dominio.

Componentes para programación numérica:

- Números de punto flotante
- Operaciones (+, -, etc)
- Relaciones(=, <, >, etc)
- La selección de frameworks determina la clase de lenguaje(imperativo o aplicativo).
- La selección de componentes orienta a la clase de problemas a los que se aplicará el lenguaje(numéricos o simbólicos).

Es posible aprender una metodología de programación aplicativa sin aprender un lenguaje de programación aplicativa particular. Por una parte, la mayoría de los lenguajes aplicativos tienen el mismo framework, basicamente es necesario definir funciones que incluyan condicionales y funciones recursivas y la llamada a estas funciones.

Por otra parte, muchos lenguajes aplicativos proveen componentes similares. Los componentes de un lenguaje aplicativo son tipos de datos propios de él y las operaciones definidas sobre valores llegan a ser de estos tipos, y estos tipos de datos pueden ser clasificados como atómicos (indivisibles) o compuestos (estructuras de datos).

a)Tipo de datos atómicos

Los tipos de datos atómicos son aquellos que son indivisibles, es decir no compuestos, por ejemplo datos boleanos, enteros, flotantes y cadenas. Para aritmética es son necesarios los tipos de datos atómicos, operadores (+, -, x, /) y las relaciones( ).

Un tipo de dato abstracto, es aquel que se denota en términos de un conjunto (abstracto) de valores y operaciones; mas que en términos de una implementación(concreta).

Usaremos la convención de que todos los tipos son abstractos, a menos que se especifique lo contrario.

b) Secuencias

Las secuencias son un tipo de dato (abstracto) compuesto porque sus valores pueden dividirse en otros mas pequeños.

Valores de las secuencias:

Definimos una secuencia como secuencias finitas de valores de un tipo, delimitados por <>:

Ejemplos:

Secuencias de enteros <1, 8, 9, 3>
Secuencias de strings <'las', 'frases', 'celebres', 'de', 'hoy'>
Secuencias de secuencias <<1, 5>, <2>, <7, 8>>
Secuencias nulas <>

c) Operaciones sobre secuencias

Para determinar las operaciones primitivas sobre secuencias que nos permitan realizar una programación completa es necesario tener presentes los requerimientos fundamentales sobre las operaciones primitivas:

- Constructor, nos permite construir alguna secuencia en un número finito de pasos.
- Selector, nos permite seleccionar un elemento de la secuencia.
- Discriminador, nos permite distinguir entre diferentes clases de secuencias.

Por simplicidad es importante tener un pequeño número de constructores, selectores y discriminadores.

Son necesarios solo dos constructores:

1. nil, con el que podemos crear una secuencia vacía, y
2. prefix, con la que podemos insertar un valor arbitrario dentro de una secuencia arbitraria.

Es natural que exista un discriminador que distinga entre las dos clases de secuencias creadas por los constructores:

null, el cual determina si el argumento es null (generado por el constructor nil).

Hay solo dos clases de secuencias para las que los selectores pueden utilizarse para recuperar componentes:

- first, el cual recupera el primer elemento de una secuencia.
- rest, el cual recupera el resto de la secuencia.

Una secuencia nula no tiene asociado un selector.

d) Arquetipo para tipo de datos abstractos

Para definir un tipo de dato abstracto debemos considerar:

- Primeramente, es necesario conocer la sintaxis de los tipos de datos: las constantes y operaciones primitivas del tipo de dato, y las reglas para combinar estos.
- Segundo, es necesario describir la semántica de los tipos de datos: el significado de las operaciones primitivas, es decir, los valores calculados cuando obtienen una entrada correcta.
- Finalmente, es necesario describir la pragmática del tipo de dato: el propósito, efectos, e implicaciones del uso actual de las operaciones primitivas.

e) Sintaxis

Permite describir la forma correcta en la que las expresiones deben ser construidas.

La sintaxis para una operación primitiva debe especificar:

1. El nombre de la operación,
2. Los tipos de los argumentos.
3. El tipo de valor que regresa.

f) Semántica

Nos permite establecer como deben interpretarse las fórmulas por la computadora. Esto se puede lograr proporcionando un conjunto de axiomas matemáticos que especifiquen los valores que pueden tomar los tipos y que determinen el valor de las formulas. Así, la especificación semántica consiste de dos partes:

1. Un axioma de existencia (nos dice que valores pueden tomar los tipos).
2. Un conjunto de ecuaciones (nos dicen que valores se obtienen de las operaciones).

g) Pragmática

Nos permite evaluar el performance de una función (nos describe una constante de tiempo que tarda una función en ejecutarse). También en esta sección podemos discutir las limitaciones permisibles sobre las implementaciones de los tipos de datos.

4.3. Introducción a LISP

LISP fue proyectado e implementado inicialmente por John McCarthy y un grupo del Massachussets Institute of Technology alrededor de 1960. El lenguaje se ha llegado a usar ampliamente para investigación en ciencias de la computación, en forma destacada en el área de inteligencia artificial (LA: robótica, procesamiento de lenguajes naturales, prueba de teoremas, sistemas inteligentes, etc.). Se han desarrollado muchas versiones de LISP a lo largo de los últimos 30 años y, de todos los lenguajes que se describen en este libro, LISP es el único que no está estandarizado ni dominado por una implementación particular.

LISP es diferente de casi todos los demás lenguajes en varios aspectos. El más notable es la equivalencia de forma entre programas y datos en el lenguaje, la cual permite ejecutar estructuras de datos como programas y modificar programas como datos. Otra característica destacada es la fuerte dependencia de la recursión como estructura de control, en vez de la iteración (formación de ciclos) que es común en casi todos los lenguajes de programación.

Historia

LISP tuvo sus inicios en el MIT alrededor de 1960. Durante los años sesenta y setenta se desarrollaron varias versiones del lenguaje. Éstas incluian el MacLisp del MIT, el Interlisp de WarrenTeitelman para la DEC PDP-10, el Spice LISP y el

Franz LISP. El Interlisp fue el dialecto dominante durante la mayor parte de esta época. Durante la parte final de la década de 1970, Gerald Sussman y Guy Steele desarrollaron una variante, como parte de una investigación sobre modelos de computación, que se conocía como “Schemer”, pero, debido a limitaciones a un nombre de seis caracteres, se abrevió a simplemente Scheme. Ésta es la versión que ha tenido el máximo impacto en el uso universitario del lenguaje.

En abril de 1981, se llevó a cabo una reunión entre las diversas facciones del LISP para tratar de fusionar los diversos dialectos en un solo lenguaje LISP, el cual se difundió con el nombre de Common LISP, a falta de un mejor nombre “común”. El nombre “Standard LISP” ya había sido adoptado por uno de estos dialectos. La estructura básica del CorumonLISP se desarrolló a lo largo de los tres años siguientes.

La época desde 1985 hasta principios de los años noventa representó probablemente la de máxima popularidad para LISP. La investigación en JA prosperaba, y en muchas universidades, quizá el 75% o más de los estudiantes de posgrado declaraban la lA como su área de especialización.

En 1986, el grupo técnico de trabajo X3J13 se reunió para estandarizar el CommonLISP,yse inició un esfuerzo por fusionar los dialectos Scheme y Common LISP. Esto fracasó, y en 1989 el IEEE desarrolló un estándar propio para Scheme. Alrededor de esta época, los efectos de la orientación a objetos en lenguajes como C++ y Smalltalk se estaban haciendo sentir, y se escribió el Common LISPObject System(CLOS; SistemadeObjetosenCommonLISP). Finalmente, en 1992, el X3J13 desarrolló un borrador para Common LISP de más de 1000 páginas, más grande que el estándar para COBOL.

Parece extraño que un lenguaje con un diseño básico tan sencillo y limpio pudiera crecer fuera de control.

Desde un principio, LISP sufrió críticas por ejecución lenta, en especial en la computadora estándar de von Neumann. Así como las funciones originales car y cdr tuvieron como modelo el hardware de la IBM 704, se estaban ideando arquitecturas de máquina altemativas para acelerar la ejecución del LISP. Varias compañías desarrollaron máquinas proyectadas para la ejecución rápida de LISP. Sin embargo, alrededor de 1989 se desarrolló una estrategia de recolección de basura para arquitecturas normales de von Neumann que era competitiva con el hardware de LISP para usos especiales. Por esta razón, ninguna de las compañías de LISP sobrevivió hasta llegar a ser un éxito comercial a largo plazo.

Ejemplo:

LISP, al igual que FORTRAN, es uno de los lenguajes de programación más antiguos.

Constituye un pilar de la comunidad dedicada a la inteligencia artificial y ha estado

evolucionando durante más de 30 años. Sin embargo, LISP ha sido siempre un lenguaje que ocupa un nicho y no es idóneo para la mayoría de las aplicaciones de computadora convencionales.

Aunque las versiones compiladas de LISP permiten que los programas en LISP se ejecuten con un poco más de eficiencia, los avances en el diseño de lenguajes, como el ML, suministran una manera más fácil de escribir programas aplicativos.

Existen numerosos dialectos de LISP, y los intentos por fusionar los formatos de Scheme y Common LISP en un lenguaje común no han tenido tanto éxito como se quisiera.

Tipos, estructuras y objetos en Lisp

a) Tipos de datos primitivos.

Los tipos primarios de objetos de datos en LISP son listas y átomos. Las definiciones de función y las listas de propiedades son tipos especiales de listas de particular importancia. También se suministran ordinariamente arreglos, números y cadenas, pero estos tipos desempeñan un papel inferior.

b) Variables y constantes.

En LISP, un átomo es el tipo elemental básico de un objeto de datos. A un átomo se le suele llamar átomo literal para distinguirlo de un número (o átomo numérico), el cual también es clasificado como átomo por casi todas las funciones en LISP.

Sintácticamente, un átomo es tan solo un identificador, una cadena de letras y dígitos que se inicia con una letra. No se suele tomar en cuenta la distinción de mayúsculas y minúsculas para identificadores. Dentro de las definiciones de función en LISP, los átomos tienen los usos ordinarios de los identificadores; se emplean como nombres de variables, nombres de funciones, nombres de parámetros formales, etcétera.

Sin embargo, un átomo en LISP no es simplemente un identificador en tiempo de ejecución. Un átomo es un objeto de datos complejo representado por una posición en la memoria, el cual contiene el descriptor de tipo para el átomo junto con un apuntador a una lista de propiedades. La lista de propiedades contiene las diversas propiedades asociadas al átomo, una de las cuales es siempre su nombre de impresión, que es la cadena que representa el átomo para entradas y salidas. Otras propiedades representan diversos enlaces para el átomo, las cuales pueden incluir la función nombrada por el átomo y otras propiedades asignadas por el programa durante la ejecución.

c) Tipos de datos estructurados.

Las listas en LISP son estructuras simples con un sólo vínculo. Cada elemento de lista contiene un apuntador a un elemento de datos y un apuntador al elemento de lista siguiente.

El último elemento de lista señala al átomo especial nil (nada) como su sucesor. Los dos apuntadores de una lista de elementos se conocen como elapuntador car y el apuntador cdr. El apuntador cdr señala al sucesor del elemento de lista. El apuntador car señala al elemento de datos.
Un elemento de lista puede contener (tener como apuntador car) un apuntador a un objeto de datos de cualquier tipo, incluso un átomo literal o numérico, o un apuntador a otra lista. Cada caso se distingue por un indicador de tipo de datos guardado en la localidad a la que se señala. Puesto que un elemento de lista puede contener un apuntador a otra lista, es posible construir estructuras de listas de complejidad arbitraria.

d) Control de secuencia.

LISP no distingue entre datos en LISP y programas en LISP, lo que conduce a un diseño sencillo para el intérprete de LISP. La ejecución de LISP es amena, comparada con la de otros lenguajes, en cuanto a que el intérprete básico se puede describir en LISP. A la función apply (aplicar) se le pasa un apuntador a la función por ejecutar, y apply interpreta la definición de esa función, empleando una segunda función eval para evaluar cada argumento.

e) Expresiones y condicionales

Un programa en LISP se compone de una sucesión de definiciones de función, donde cada función es una expresión en notación polaca de Cambridge. Condicionales. La condicional es la estructura principal para proporcionar una secuenciación alternativa de ejecución en un programa en LISP. La sintaxis es: donde cada alternativa, es (predicado expresión,). cond se ejecuta evaluando cada predicado,, por turno, y evaluando la expresión, del primero que devuelva cierto (7’). Si todos los predicados son falsos, se evalúa expresión por omisión.

f) Enunciados

LISP carece de un concepto real de ejecución de enunciados. Este lenguaje se ejecuta evaluando funciones. El grupo de llamadas de función normales representa el conjunto usual de “estructuras de control”. La expresión condes prácticamente el único mecanismo de secuenciación de ejecución de primitivas que se necesita.

La función prog proporciona los medios para la ejecución secuencial, pero no es absolutamente necesaria, aunque facilita la escritura de casi cualquier programa en LISP.

Una prog tiene la sintaxis:

Donde variables es una lista de variables que es conocida dentro de la sucesión de expresiones, y cada expresión, se ejecuta en orden. (progn(expresión1)...) es similar, sin las variables locales. Se puede salir de una prog ya sea completando la evaluación de la última expresión (en cuyo caso la prog tiene en conjunto el valor ni!) o por una llamada a la primitiva return. El argumento para return es el valor que debe ser devuelto como valor de la prog.

g) Entrada y salida

Las entradas y salidas son simplemente llamadas de función. (read) lee el átomo siguiente proveniente del teclado.

h) Definición de funciones

La función defun se usa para crear nuevas funciones. La sintaxis de esta función es:

donde argumentos son los parámetros para la función, y el cuerpo de la misma es una sola expresión (que puede ser una sucesión de progn u otras secuencias de ejecución complejas).

Muchos sistemas en LISP reservan una localidad especial en el átomo (bloque de cabecera) para la definición de la función que se nombra con ese átomo, para evitar la necesidad de buscar la definición de la función en la lista de propiedades cada vez que se llama la función. La localidad especial designa el tipo de función (expr,fexpr, etc.) y contiene un apuntador a la estructura de listas que representa su definición de función. En estas implementaciones, por lo común se suministran primitivas especiales para obtener, insertar y eliminar directamente una definición de función de la localidad especial.

4.4. Introducción a ML

ML es un lenguaje aplicativo con programas escritos en el estilo de C o Pascal. Sin embargo, es un lenguaje aplicativo con un concepto avanzado del tipo de datos. ML maneja polimorfismo y, a través de su sistema de tipos, maneja abstracciones de datos. El lenguaje base es relativamente compacto, en especial si se compara con la definición de un lenguaje como Ada. Sin embargo, la capacidad de ampliación de sus tipos le confiere un gran poder para desarrollar programas complejos.

Incluye excepciones, programación imperativa y funcional, especificaciones con base en reglas y casi todos los conceptos que se han presentado en otros lenguajes de este libro.

Si uno estuviera limitado a un lenguaje en el cual estudiar muchos conceptos de lenguajes, entonces ML sería una opción razonable, en tanto a uno no le importara la viabilidad comercial del lenguaje.

ML ha logrado avances significativos en las comunidades educativas y de investigación en ciencias de la computación. La exposición del mecanismo de tipos en nivel de lenguaje fuente es una característica que no está disponible en otros lenguajes ampliamente utilizados. Sin embargo, las aplicaciones comerciales de los programas en ML son pocas, y hasta ahora continúa principalmente como un vehículo para investigación en ciencias de la computación y uso educativo.

Historia

ML, por las siglas de MetaLanguage, fue desarrollado por Robin Miller, junto con otras personas, como un mecanismo para pruebas formales asistidas por computadora en el sistema de Lógica para Funciones Computables de Edimburgo desarrollado a mediados de los años setenta. Sin embargo, también se le encontró utilidad como lenguaje general para manipulación de símbolos. En 1983, el lenguaje se rediseñó y se amplió con conceptos como los módulos para convertirse en el Standard ML. Aunque por lo común se implementa como un intérprete, el ML se puede compilar con relativa facilidad. Los primeros compiladores aparecieron en 1984.

El uso del ML estándar se extendió a través de la comunidad dedicada a la investigación en lenguajes de programación durante los años finales de la década de 1980.

David Appel, de la Princeton University, y David MacQueen de los Bell Telephone Laboratories de AT&T desarrollaron una versión popular.

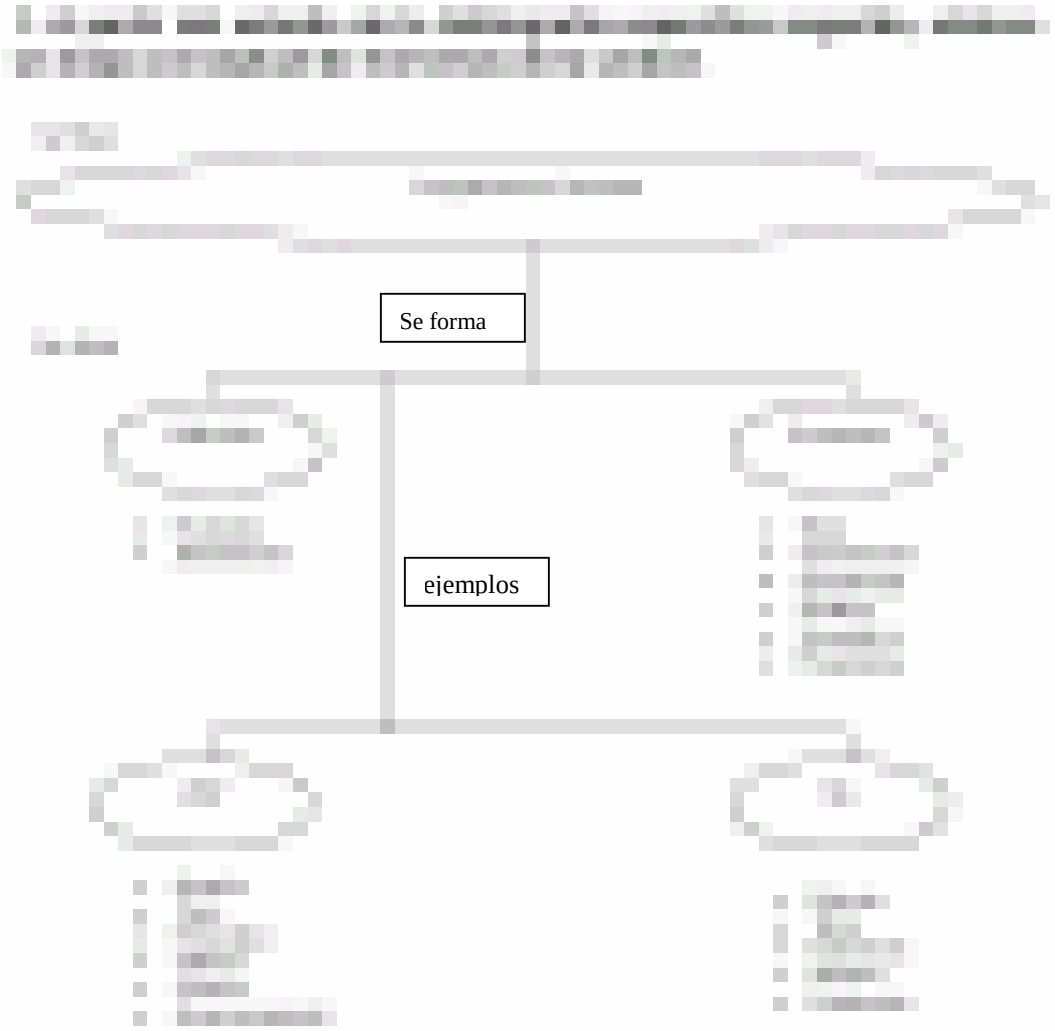
Ejemplo:

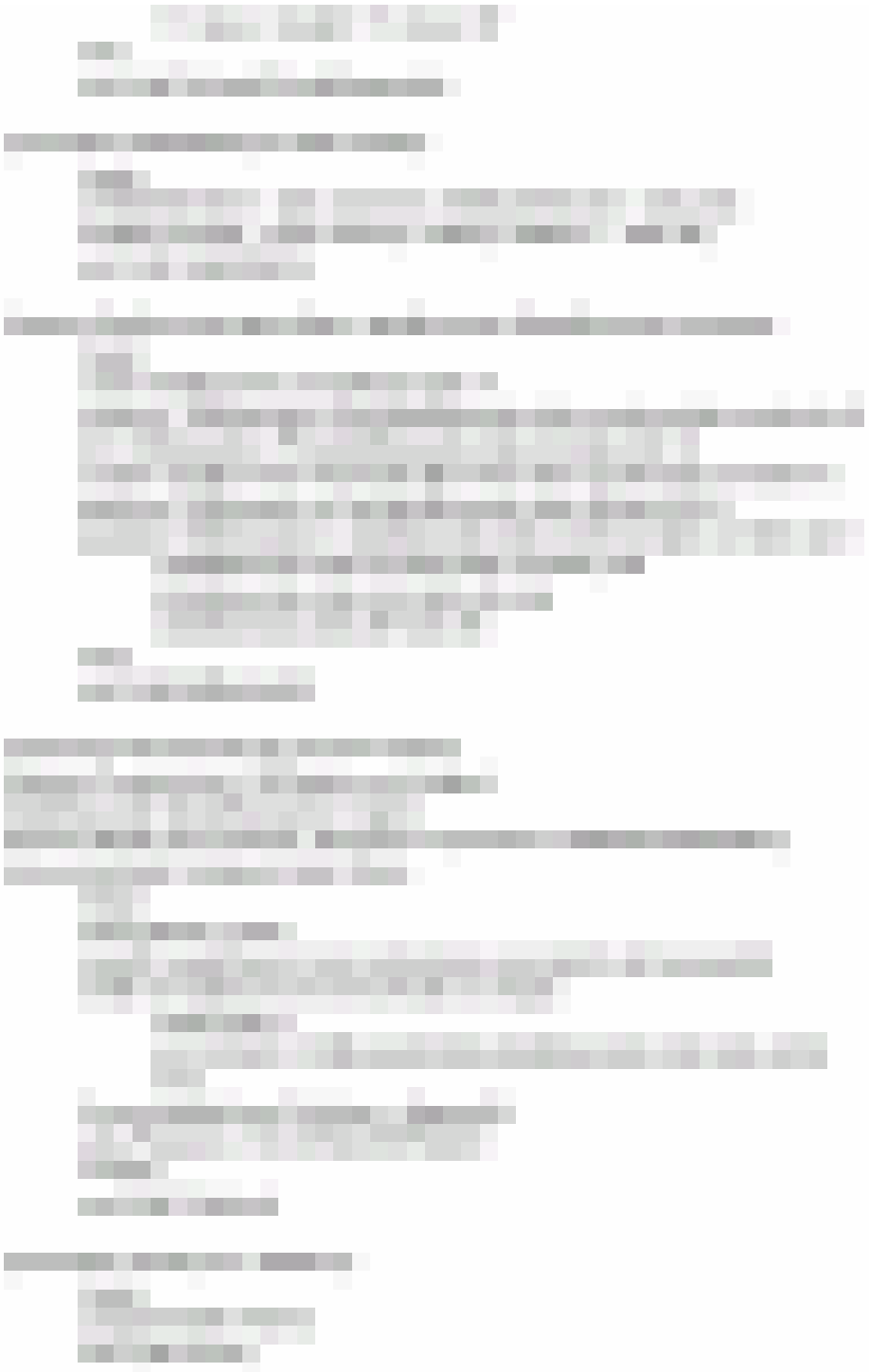
ML es un lenguaje con tipos estáticos, tipificación fuerte y ejecución aplicativa de programas. Sin embargo, difiere de otros lenguajes que hemos estudiado en cuanto a que el programador no necesita especificar los tipos. Existe un mecanismo de inferencia de tipos que determina los tipos de las expresiones resultantes. Esta inferencia de tipos da cabida a homonimia y concordancia de patrones usando unificación, en forma muy parecida a como se utiliza en Prolog.

Al igual que en LISP, un programa en ML se compone de varias definiciones de función. Cada función se tipifica estáticamente y puede devolver valores de cualquier tipo.

Puesto que es aplicativo, el almacenamiento variable se maneja en forma diferente respecto a lenguajes como C o FORTRAN. Se tiene sólo una forma limitada de asignación y, a causa de la naturaleza funcional de la ejecución, todos

los parámetros para funciones son de llamada por valor y ML crea una nueva copia de cualquier objeto complejo usando un almacén de montículo.





```
case opcion of
        '1': Crear(miAgenda);
        '2': Incorporar(miAgenda);
        '3': Consultar(miAgenda);
        '4': Borrar(miAgenda);
        end;
until opcion='4';
end.
```

Bibliografía

"Lenguajes de programación"
Diseño e implementación
Pratt – Zelkowitz
Ed.Prentice may

"Sistemas de Información para la Administración"
James A. Senn
Ed. Grupo Editorial Iberoamérica,
México, 1990

"Programing language concepts"
Ghezzi Carlo,
John Wiley and Sons
U.S.A., 1976

5. Programación Lógica

Introducción

En Octubre de 1981, el gobierno japonés y más concretamente el Ministerio Japonés de Comercio Internacional e Industria (MITI), anuncia la puesta en marcha de un proyecto revolucionario equiparable a la carrera del espacio norteamericana. Están dispuestos a ofrecer al mundo la siguiente generación, la Quinta Generación de Ordenadores. Unas máquinas de Inteligencia Artificial que pueden pensar, sacar conclusiones, emitir juicios e incluso comprender las palabras escritas y habladas.

Con este fin se crea el ICOT (Institute for New Generation Computer Technology) constituido por cuarenta brillantes investigadores de las más importantes empresas, y se les dota con todos los medios necesarios para construir la nueva clase de supercomputadoras.

La Quinta Generación prevé máquinas diseñadas para el tratamiento lógico, de capacidades análogas a las capacidades de anteriores generaciones de ordenadores para tratar operaciones aritméticas. Se trata de ordenadores que tienen el PROLOG como lenguaje nativo (lenguaje máquina), con capacidad para procesar millones de inferencias lógicas por segundo (LIPS).

PROLOG puede encontrarse en múltiples versiones diferentes. La más popular es la definida por Clocksin y Mellish, y es la más común. Afortunadamente, al ser tan sencilla la sintaxis del PROLOG, las variaciones entre distintas implementaciones son mínimas

Definición

La programación lógica es un paradigma de programación declarativa, que emerge de la fusión de tres campos:

1) La lógica de primer orden de la Lógica Matemática.
2) La demostración automática de teoremas dentro de la Inteligencia Artificial.
3) El estudio de los lenguajes formales como parte de las ciencias de la computación.

La programación lógica los programas se consideran como una serie de aserciones lógicas. De esta forma, el conocimiento se representa mediante

reglas, tratándose de sistemas declarativos. Una representación declarativa es aquella en la que el conocimiento está especificado, pero en la que la manera en que dicho conocimiento debe ser usado no viene dado. El más popular de los sistemas de programación lógica es el PROLOG.

La programación Lógica está basada en la noción de ***Relación***. Debido a que en la relación es un concepto mas general de una aplicación. La programación lógica es potencialmente de alto nivel.

Paradigma lógico

• *Programa*: conjunto de hechos y reglas.
• *Sistema*: demostrador de teoremas mediante un mecanismo de inferencia.
• *Caracterización de propiedades y relaciones.*
• *Doble dirección (E/S) de los datos.*
• *Datos parcialmente construidos.*
• *Teóricamente, el orden no importa.*

Características:

- sintáxis y semántica bien definidas
- reglas de inferencia

Introduccion a PROLOG

Historia

El desarrollo de Prolog se inició en 1970 con Alain Coulmerauer y Philippe Roussel, quienes estaban interesados en desarrollar un lenguaje para hacer deducciones a partir de texto. El nombre corresponde a “PROgramming in LOGic” (Programación en lógica). Prolog fue desarrollado en Marsella, Francia, en 1972. El principio de resolución de Kowalski, de la Universidad de Edimburgo pareció un modelo apropiado para desarrollar sobre él un mecanismo de inferencia.

Con la limitación de la resolución a cláusulas de Horn, la unificación condujo a un sistema eficaz donde el no determinismo inherente de la resolución se manejó por medio de un proceso de exploración a la inversa, el cual se podía implementar con facilidad. El algoritmo para resolución suministró la secuencia

de ejecución que se requería para implementar los ejemplos de especificaciones como la relación de vuelos antes señalada.

La primera implementación de Prolog se completó en 1972 usando el compilador de ALGOL W de Wirth, y los aspectos básicos del lenguaje actual se concluyeron para 1973. El uso del Prolog se extendió gradualmente entre quienes se dedicaban a la programación lógica principalmente por contacto personal y no a través de una comercialización del producto. Existen varias versiones diferentes, aunque bastante similares. Aunque no hay un estándar del Prolog, la versión desarrollada en la Universidad de Edimburgo ha llegado a ser utilizada ampliamente. El uso de este lenguaje no se extendió sino hasta los años ochenta.

La falta de desarrollo de aplicaciones eficaces de Prolog inhibieron su difusión.

Definición

Prolog es un lenguaje de programación seminterpretado. Su funcionamiento es muy similar a Java. El codigo fuente se compila a un código de byte el cuál se interpreta en una máquina virtual denominada Warren Abstract Machine (comúnmente denominada WAM).

PROLOG es un lenguaje de programación para ordenadores que se basa en el lenguaje de la Lógica de Primer Orden y que se utiliza para resolver problemas en los que entran en juego *objetos* y *relaciones* entre ellos. Por ejemplo, cuando decimos "Jorge tiene una moto", estamos expresando una relación entre un objeto (Jorge) y otro objeto en particular (una moto). Más aún, estas relaciones tienen un orden específico (Jorge posee la moto y no al contrario). Por otra parte, cuando realizamos una pregunta (¿Tiene Jorge una moto?) lo que estamos haciendo es indagando acerca de una relación.

Además, también solemos usar reglas para describir relaciones: "dos personas son hermanas si ambas son hembras y tienen los mismos padres". Como veremos más adelante, esto es lo que hacemos en Prolog.

Una de las ventajas de la programación lógica es que se especifica *qué* se tiene que hacer *(programación declarativ*a), y no *cómo* se debe hacer *(programación imperativ*a). A pesar de esto, Prolog incluye algunos predicados predefinidos meta-lógicos, ajenos al ámbito de la Lógica de Primer Orden, (var, nonvar, ==, ...), otros extra-lógicos, que tienen un efecto lateral, (write, get, ...) y un tercer grupo que nos sirven para expresar información de control de como realizar alguna tarea (el corte, ...).

Por tanto, Prolog ofrece un sistema de programación práctico que tiene algunas de las ventajas de claridad y declaratividad que ofrecería un lenguaje de

programación lógica y, al mismo tiempo, nos permite un cierto control y operatividad.

Prolog representa el lenguaje principal en la categoría de programación lógica. A diferencia de otros lenguajes no es un lenguaje de programación para usos generales, sino que está orientado a resolver problemas usando el cálculo de predicados.

Aplicaciones del Prolog

Provienen en general de dos dominios distintos:

1) Preguntas a bases de datos. Las bases de datos modernas indican típicamente relaciones entre los elementos que están guardados en la base de datos. Pero no todas estas relaciones se pueden indicar. Por ejemplo, en una base de datos de una línea aérea, puede haber entradas que indican números de vuelo, ubicación y hora de salida, y ubicación y hora de llegada. Sin embargo, si un individuo necesita hacer un viaje que requiera cambiar de avión en algún punto intermedio, es probable que esa relación no esté especificada en forma explícita.

2) Pruebas matemáticas. También se pueden especificar las relaciones entre objetos matemáticos a través de una serie de reglas y sería deseable un mecanismo para generar pruebas de teoremas a partir de este modelo. Aunque la búsqueda ascendente inherente en LISP da cabida a la construcción de sistemas generadores de pruebas, sería eficaz un lenguaje orientado en mayor grado hacia la prueba de propiedades de relaciones.

Estas dos aplicaciones son similares y se pueden resolver usando Prolog. El objetivo para Prolog era proporcionar las especificaciones de una solución y permitir que la computadora dedujera la secuencia de ejecución para esa solución, en vez de especificar un algoritmo para la solución de un problema, como es el caso normal de casi todos los lenguajes que hemos estudiado.

Por ejemplo, si se tiene información de vuelos de una línea aérea de la forma: vuelo (número_de_vuelo ciudad_origen, ciudad_destino hora_de_salida hora_de llegada) entonces todos los vuelos de Los Ángeles a Baltimore se pueden especificar ya sea como vuelos directos: vuelo(número_de_vuelo, Los Ángeles, Baltimore hora_de_salida hora de llegada) o como vuelos con una parada intermedia, especificados como:

vuelo(vuelol, Los Ángeles, X, salel, llegal),
vuelo(vuelo2, X,Baltimore, sale2, llega2),
sale2>= llegal+30

lo que indica que se está especificando una ciudad X, a la cual llega un vuelo proveniente de Los Ángeles, y de la que sale un vuelo para Baltimore, y el vuelo con destino a Baltimore sale al menos 30 minutos después de que llega el vuelo de los Ángeles para dar tiempo a cambiar de avión. No se especifica un algoritmo; sólo se han indicado las condiciones para tener una solución correcta.

El lenguaje mismo, si se puede enunciar un conjunto de condiciones de esta naturaleza, suministrará la secuencia de ejecución que se necesita para encontrar el vuelo apropiado.

Un entorno de desarrollo Prolog se compone de:

- **Un compilador**. Transforma el código fuente en código de byte. A diferencia de Java, no existe un standard al respecto. Por eso, el codigo de byte generado por un entorno de desarrollo no tiene por que funcionar en el intérprete de otro entorno.

- **Un intérprete**. Ejecuta el código de byte.

- **Un shell o top-level**. Se trata de una utilidad que permite probar los programas, depurarlos, etc. Su funcionamiento es similar a los interfaces de línea de comando de los sistemas operativos.

- **Una biblioteca de utilidades**. Estas bibliotecas son, en general, muy amplias. Muchos entornos incluyen (afortunadamente) unas bibliotecas standard-ISO que permiten funcionalidades básicas como manipular cadenas, entrada/salida, etc.

Prolog es aproximadamente diez veces más lento que el lenguaje C. Pero también hemos de admitir que un programa en Prolog ocupa aproximadamente diez veces menos, en líneas de código y tiempo de desarrollo, que el mismo programa escrito en C. Además las técnicas de optimización de código en

Objetos de datos

a) Tipos de datos primitivos

Variables y constantes. Los datos en Prolog incluyen enteros (1,2,3, ...), reales (1.2,3.4, 5.4, ...) y caracteres (‘a’, ‘b’, ...). Los nombres que comienzan con letra minúscula representan hechos concretos, si son parte de un hecho de la base de datos, o, si están escritos con mayúsculas, representan variables que se pueden unificar con hechos durante la ejecución de Prolog. El alcance de una variable es la regla dentro de la cual se usa.

b) Tipos de datos estructurados

Los objetos pueden ser átomos (constantes y variables de cadena) o listas. Una lista se representa como [A,B,C,..jJ. La notación [AIB] se usa para indicar A como cabeza de la lista y B como cola de la lista:

La cadena ‘abcd’ representa la lista [‘a’, ‘b’, ‘c’, ‘d’]. La variable _ es una variable arbitraria sin nombre. Así, [A l j hace coincidir A con la cabeza de cualquier lista.

c) Tipos definidos por el usuario

No existen verdaderos tipos definidos por el usuario; sin embargo, las reglas para definir relaciones pueden actuar como si fueran tipos de usuario.

Definiciones y conceptos:

Hechos

Expresan relaciones entre objetos. Supongamos que queremos expresar el hecho de que "un coche tiene ruedas". Este hecho, consta de dos objetos, "coche" y "ruedas", y de una relación llamada "tiene". La forma de representarlo en PROLOG es:

tiene(coche,ruedas).

- Los nombres de objetos y relaciones deben comenzar con una letra minúscula.
- Primero se escribe la relación, y luego los objetos separados por comas y encerrados entre paréntesis.
- Al final de un hecho debe ir un punto (el caracter ".").
- El orden de los objetos dentro de la relación es arbitrario, pero debemos ser coherentes a lo largo de la base de hechos.

Variables

Representan objetos que el mismo PROLOG determina . Una variable puede estar instanciada ó no instanciada. Estar instanciada cuando existe un objeto determinado representado por la variable. De este modo, cuando preguntamos "¨

Un coche tiene X ?",

PROLOG busca en los hechos cosas que tiene un coche y respondería:

X = ruedas.

instanciando la variable X con el objeto ruedas.

Los nombres de variables comienzan siempre por una letra mayúscula. Un caso particular es la variable anónima, representada por el carácter subrayado ("_").

Es una especie de comodín que utilizaremos en aquellos lugares que debería aparecer una variable, pero no nos interesa darle un nombre concreto ya que no vamos a utilizarla posteriormente.

El ámbito de las variables. Cuando en una regla aparece una variable, el ámbito de esa variable es únicamente esa regla. Supongamos las siguientes reglas:

(1) hermana_de(X,Y) :- hembra(X), padres(X,M,P), padres(Y,M,P).
(2) puede_robar(X,P) :- ladron(X), le_gusta_a(X,P), valioso(P).

Aunque en ambas aparece la variable X (y la variable P), no tiene nada que ver la X de la regla (1) con la de la regla (2), y por lo tanto, la instanciación de la X en (1) no implica la instanciacion en (2). Sin embargo todas las X de *una misma regla* sí que se instanciar n con el mismo valor.

Reglas

Las reglas se utilizan en PROLOG para significar que un hecho depende de uno ó mas hechos. Son la representación de las implicaciones lógicas del tipo p ---> q (p implica q).

- Una regla consiste en una cabeza y un cuerpo, unidos por el signo ":-".
- La cabeza está formada por un único hecho.
- El cuerpo puede ser uno o más hechos (conjunción de hechos), separados por una coma (","), que actúa como el "y" lógico.
-Las reglas finalizan con un punto (".").

La cabeza en una regla PROLOG corresponde al consecuente de una implicación lógica, y el cuerpo al antecedente. Este hecho puede conducir a errores de representación.

Supongamos el siguiente razonamiento lógico:

tiempo(lluvioso) ----> suelo(mojado)
suelo(mojado)

Que el suelo esté mojado, es una condición suficiente de que el tiempo sea lluvioso, pero no necesaria. Por lo tanto, a partir de ese hecho, no podemos deducir mediante la implicación, que esté lloviendo (pueden haber regado las calles). La representación *correcta* en PROLOG, sería:

suelo(mojado) :- tiempo(lluvioso).
suelo(mojado).

Adviértase que la regla está "al revés". Esto es así por el mecanismo de deducción hacia atrás que emplea PROLOG. Si cometiéramos el *error* de representarla como:

tiempo(lluvioso) :- suelo(mojado).
suelo(mojado).

PROLOG, partiendo del hecho de que el suelo está mojado, deduciría incorrectamente que el tiempo es lluvioso.

Para generalizar una relación entre objetos mediante una regla, utilizaremos variables.

Por ejemplo:

Representación lógica | Representación PROLOG
----------+----------
es_un_coche(X) ----> | tiene(X,ruedas) :-
tiene(X,ruedas) | es_un_coche(X).

Con esta regla generalizamos el hecho de que cualquier objeto que sea un coche, tendrá ruedas. Al igual que antes, el hecho de que un objeto tenga ruedas, no es una condición suficiente de que sea un coche. Por lo tanto la representación inversa sería incorrecta.

Operadores

Son predicados predefinidos en PROLOG para las operaciones matemáticas básicas.

Su sintaxis depende de la posición que ocupen, pudiendo ser infijos ó prefijos.

Por ejemplo el operador suma ("+"), podemos encontrarlo en forma prefija '+(2,5)' ó bien infija, '2 + 5'.

También disponemos de predicados de igualdad y desigualdad.

X = Y igual
X \= Y distinto
X < Y menor
X > Y mayor
X =< Y menor ó igual
X >= Y mayor ó igual

Al igual que en otros lenguajes de programación es necesario tener en cuenta la precedencia y la asociatividad de los operadores antes de trabajar con ellos.

En cuanto a precedencia, es la típica. Por ejemplo, 3+2*6 se evalúa como 3+(2*6).

En lo referente a la asociatividad, PROLOG es asociativo por la izquierda. Así, 8/4/4 se interpreta como (8/4)/4. De igual forma, 5+8/2/2 significa 5+((8/2)/2).

El operador 'is'. Es un operador infijo, que en su parte derecha lleva un término que se interpreta como una expresión aritmética, contrastándose con el término de su izquierda. Por ejemplo, la expresión '6 is 4+3.' es falsa. Por otra parte, si la expresión es 'X is 4+3.', el resultado ser la instanciación de X:

X = 7

Una regla PROLOG puede ser esta:

densidad(X,Y) :- poblacion(X,P), area(X,A), Y is P/A.

Estilo de programación en PROLOG

Los lenguajes de Programación Lógica, al igual que los lenguajes procedimentales, necesitan de una metodología para construir y mantener programas largos, así como un buen estilo de programación. Prolog ofrece mecanismos de control típicos de cualquier lenguaje imperativo y no puras propiedades o relaciones entre objetos. Dichos mecanismos contribuyen a dar mayor potencia y expresividad al lenguaje, pero violan su naturaleza lógica.

Adoptando un estilo apropiado de programación podemos beneficiarnos de esta doble naturaleza de Prolog.

- *Metodología de Diseño "top-down*": descomponemos el problema en subproblemas y los resolvemos. Para ello solucionamos primero los pequeños problemas.

- Una vez analizado el problema y separado en trozos, el siguiente paso es decidir como representar y manipular tanto los objetos como las relaciones del problema. Debemos elegir los nombres de los predicados, variables, constantes y estructuras de manera que aporten claridad a nuestros programas (palabras o nombres mnemónicos que estén relacionados con lo que hacen).

- El siguiente paso es asegurarse de que la organización y la sintaxis del programa sea clara y fácilmente legible.

- Llamamos *procedimiento* al conjunto de cláusulas para un predicado dado. Agruparemos por bloques los procedimientos de un mismo predicado (mismo nombre y mismo número de argumentos). Así, cada cláusula de un procedimiento comenzará en una línea nueva, y dejaremos una línea en blanco entre procedimientos.

- Si el cuerpo de una regla es lo bastante corto, lo pondremos en una línea; sino, se escriben los objetivos de las conjunciones indentados y en líneas separadas.

- Primero se escriben los hechos, y luego las reglas.

- Es recomendable añadir comentarios y utilizar espacios y líneas en blanco que hagan el programa más legible. El listado del programa debe estar "autodocumentado". Debemos incluir para cada procedimiento y antes de las cláusulas el *esquema de la relació*n.

- Los argumentos deben ir precedidos por el signos `+', `-' o `?'. `+' indica que el argumento es de entrada al predicado (debe estar instanciado cuando se hace la llamada), `-' denota que es de salida y `?' indica que puede ser tanto de entrada como de salida.

- Agrupar los términos adecuadamente. Dividir el programa en partes razonablemente autocontenidas (por ejemplo, todos los procedimientos de procesado de listas en un mismo fichero).

- Evitar el uso excesivo del corte.

- Cuidar el orden de las cláusulas de un procedimiento. Siempre que sea posible escribiremos las condiciones limite de la recursividad antes de las demás cláusulas. Las cláusulas "recogetodo" tienen que ponerse al final.

Subprogramas y gestión de almacenamiento

Prolog tiene dos modos: modo de consulta y modo de pregunta. En el modo de consulta se introducen nuevas relaciones en el almacenamiento dinámico de la base de datos. Durante el modo de pregunta, se ejecuta un intérprete simplemente basado en pilas para evaluar preguntas.

Ambiente local de referencia. Todos los nombres de variables son locales para la regla en la cual están definidos. La unificación prevé la interacción de nombres locales de una regla con los nombres de otra regla.

Ambiente común de referencia. Todos los datos son compartidos. No existe un verdadero concepto de ambiente local o global de referencia.

Paso de parámetros. La unificación prevé el paso de argumentos entre reglas.

Funciones normales. Casi todos los sistemas en Prolog incluyen varias funciones integradas para ayudar en la generación de programas:

consult(nombredearchivo) lee el archivo nombredearchivo y anexa nuevos hechos y reglas a la base de datos. nombredearchivo también se puede escribir como ‘nombredearchiv ‘si el nombre del archivo. contiene caracteres incrustados no pertenecientes a un identificador. consult(nombredearchivo) se suele poder especificar simplemente como
[nombredearchivo].

· reconsult(nombredearchivo) sobreescribe relaciones en la base de datos.
· fail siempre fracasa.
· see(nombredearchivo) lee entradas desde nombredearchivo como una serie de reglas.
· write(término) escribe término.
· tell(nombredearchivo) reorienta la salida de write a nombredearchivo.
· told cierra el archivo respecto a una tell previa y reorienta la salida de write a la terminal normal de visualización.
· nl pasa al renglón siguiente (de entradas y salidas).
· atom(X) es un predicado que devuelve cierto si X es un átomo (constantes de cadena o nombres de variable).
· var(X) es un predicado que devuelve cierto si Xes una variable.
· integer(X) es un predicado que devuelve cierto si Xes un entero.
· trace activa la depuración porrastreo mostrando cada paso de la ejecución. notrace la desactiva.

Evaluación del lenguaje

Prolog ha alcanzado una medida razonable de éxito como lenguaje para resolver problemas de relaciones, como en el procesamiento de consultas a bases de datos. Ha alcanzado un éxito limitado en otros dominios.

El objetivo original de poder especificar un programa sin proporcionar sus detalles algorítmicos no se ha alcanzado realmente. Los programas en Prolog "se leen secuencialmente", aunque el desarrollo de reglas sigue un estilo aplicativo. Se usan cortes con frecuencia para limitar el espacio de búsqueda para una regla, con el efecto de hacer el programa tan lineal en cuanto a ejecución como un programa típico en C o Pascal. Aunque las reglas se suelen expresar en forma aplicativa, el lado derecho de cada regla se procesa de manera secuencial. Esto sigue un patrón muy similar al de los programas en ML.

Perspectiva del lenguaje

Un programa en Prolog se compone de una serie de hechos, relaciones concretas entre objetos de datos (hechos) y un conjunto de reglas, es decir, un patrón de relaciones entre los objetos de la base de datos. Estos hechos y reglas se introducen en la base de datos a través de una operación de consulta.

Un programa se ejecuta cuando el usuario introduce una pregunta, un conjunto de términos que deben ser todos ciertos. Los hechos y reglas de la base de datos se usan para determinar cuáles sustituciones de variables de la pregunta (llamadas unificación) son congruentes con la información de la base de datos.

Como intérprete, Prolog solicita entradas al usuario. El usuario digita una pregunta o un nombre de función. La verdad ("yes") o falsedad ("no") de esa preguntase imprime, así como una asignación a las variables de la pregunta que hacen cierta la pregunta, es decir , que unifican la pregunta. Si se introduce un ";", entonces se imprime el próximo conjunto de valores que unifican la pregunta, hasta que no son posibles más sustituciones, momento en el que Prolog imprime "no" y aguarda una nueva pregunta. Un cambio de renglón se interpreta como terminación de la búsqueda de soluciones adicionales.

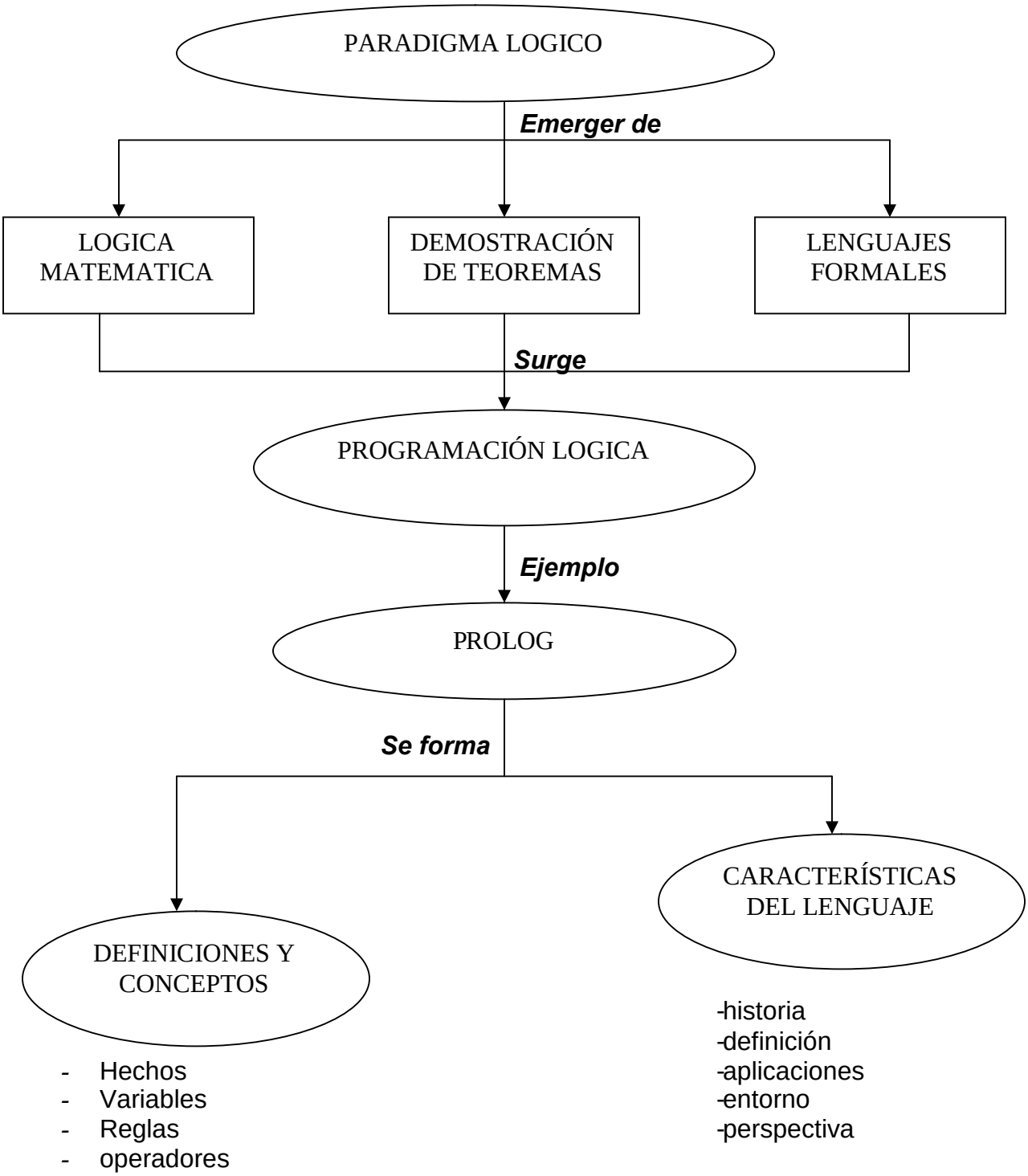
La ejecución de Prolog, aunque se basa en la especificación de predicados, opera en forma muy parecida a un lenguaje aplicativo como LISP o ML. El desarrollo de reglas en Prolog requiere el mismo "pensamiento recursivo" que se necesita para desarrollar programas en esos otros lenguajes aplicativos.

Prolog tiene una sintaxis y semántica simples. Puesto que busca relaciones entre una serie de objetos, la variable y la lista son las estructuras de datos básicas que se usan. Una regla se comporta en forma muy parecida a un procedimiento, excepto que el concepto de unificación es más complejo que el proceso relativamente sencillo de sustitución de parámetros por expresiones.

Prolog emplea la misma sintaxis que C y PL/I para comentarios: /* ... */.

ACTIVIDADES COMPLEMENTARIAS

1- A partir del estudio de la bibliografía sugerida, elabore un mapa conceptual de los temas de la unidad



2. Ejemplo de programa en PROLOG:

```
quiere_a(maria,enrique).

quiere_a(juan,jorge).

quiere_a(maria,susana).

quiere_a(maria,ana).

quiere_a(susana,pablo).

quiere_a(ana,jorge).

varon(juan).

varon(pablo).

varon(jorge).

varon(enrique).

hembra(maria).

hembra(susana).

hembra(ana).

teme_a(susana,pablo).

teme_a(jorge,enrique).

teme_a(maria,pablo).

/* Esta linea es un comentario */

quiere_pero_teme_a(X,Y) :- quiere_a(X,Y), teme_a(X,Y).

querido_por(X,Y) :- quiere_a(Y,X).

puede_casarse_con(X,Y) :- quiere_a(X,Y), varon(X), hembra(Y).

puede_casarse_con(X,Y) :- quiere_a(X,Y), hembra(X), varon(Y).
```

A. Bibliografía

“Lenguajes de programación” Diseño e implementación
Pratt – Zelkowitz
Ed.Prentice may

“Sistemas de Información para la Administración”
James A. Senn
Ed. Grupo Editorial Iberoamérica,
México, 1990

“Enciclopedia Encarta 99”
Microsoft

UNIDAD 6: Matriz comparativa de los paradigmas

Análisis comparativo de los diferentes paradigmas de la programación

PARADIGMAS	CARACTERÍSTICAS	LENGUAJE EJEMPLO
IMPERATIVO Los lenguajes imperativos son lenguajes controlados por mandatos u orientados a enunciados (instrucciones).	- Comandos o Instrucciones. - Orientados a la utilización por programadores profesionales. - Requiere especificación sobre cómo ejecutar una tarea. - Se deben especificar todas las alternativas. - Requiere gran número de instrucciones de procedimiento. - El código puede ser difícil de leer, entender y mantener. - Lenguaje creado originalmente para operación por lotes. - Puede ser difícil de aprender. - Difícil de depurar. - Orientados comúnmente a archivos.	• Fortran (FORmula TRANslation) • Basic (Beglnner's All-purpose Symbolic Instruction Code) • Cobol (COmmon Business-Oriented Language) • Pascal • Ada • ALGOL • Smalltalk • PL/l • C
ORIENTADA A OBJETOS La idea principal de la programación orientada a objetos es construir programas que utilizan objetos de software. Un objeto puede considerarse como una entidad independientemente de cómputo con sus propios datos y programación.	-Se basa en conceptos sencillos: objetos y atributos, el todo y las partes, clases y miembros. - Encapsulación - Herencia - Polimorfismo - Facilita la creación de prototipos - Simplicidad - Modularidad -Facilidad para hacer modificaciones - Posibilidad de extenderlo	• Java • PHP • ASP • PERL • BASH • Visual J ++
FUNCIONAL La programación funcional se basa en el uso de las funciones matemáticas para producir mejores efectos y que sus resultados sean más eficientes.	- Semánticas claras, simples y matemáticamente bien fundadas - Cercanos al nivel de abstracción de las especificaciones formales / informales de los problemas a resolver - Referencialmente transparentes: Comportamiento matemático adecuado que permite razonar sobre los programas -Soportan técnicas muy	• ML (MetaLanguage) • LISP

	avanzadas de desarrollo, mantenimiento y validación de programas -Altas dosis de paralelismo implícito - Aplicaciones variadas y de gran interés - Provee de un paradigma para programar en paralelo. - Es ampliamente aplicada en la Inteligencia Artificial. - Es bastante útil para el desarrollo de prototipos.	
LOGICA La programación lógica los programas se consideran como una serie de aserciones lógicas. De esta forma, el conocimiento se representa mediante reglas, tratándose de sistemas declarativos	- Emerge de la lógica de primer orden de la Lógica Matemática. - Busca la demostración automática de teoremas dentro de la Inteligencia Artificial, mediante un mecanismo de inferencia. - Se basa en el estudio de los lenguajes formales como parte de las ciencias de la computación. - Caracterización de propiedades y relaciones. - Doble dirección (E/S) de los datos. - Datos parcialmente construidos. -Teóricamente, el orden no importa. -Sintáxis y semántica bien definidas -Reglas de inferencia	• PROLOG

UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO.

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN.

SISTEMA DE UNIVERSIDAD ABIERTA.

INFORMATICA IV.

UNIDAD VI.

ANÁLISIS COMPARATIVO DE LOS DIFERENTES PARADIGMAS DE LA PROGRAMACIÓN.

LIC. VICENTE HERNANDEZ.

6.1 DEFINICION DE PARADIGMA.

Según el Diccionario El Pequeño Larousse Ilustrado, edición 1999, en su pagina 759 anota que Paradigma viene de la palabra griega *paradeigma:* Ejemplo que sirve de norma. En la Filosofía Platónica: El mundo de las ideas, Prototipo del mundo sensible que vivimos.
Como se puede apreciar es un ejemplo a seguir por cuanto a lo valioso que es y representa.
El Paradigma en el ambiente de programación es la plataforma teórica-conceptual a través de la cual, el planteamiento de un problema se puede resolver relacionando sus elementos con características diferentes como son la definición y tipo de los datos, la arquitectura del compilador, entre otros elementos.

6.2 PARADIGMAS DE ALGUNOS LENGUAJES DE PROGRAMACIÓN.

Aunado a los matices del pensamiento para resolver los problemas a través de un algoritmo se agregan las restricciones de la arquitectura de la computadora, la arquitectura del compilador, los tiempos de ejecución, la eficiencia del empleo del sistema en general para combinar posteriormente combinar estos elementos para investigar la creación de nuevas plataformas teóricas en el manejo de las variables, su definición, la definición de la estructura de datos, la técnica recursiva, y eventualmente encontrar un estilo nuevo en la solución de problemas. La concepción de paradigmas no solo trasciende en la búsqueda de configuraciones más eficientes sino también en el estudio teórico de la computación y consecuentemente encontrar un modelo representativo de la realidad.

PARADIGMAS	CARACTERÍSTICAS	LENGUAJE EJEMPLO
IMPERATIVO Los lenguajes imperativos son lenguajes controlados por mandatos u orientados a enunciados (instrucciones).	- Comandos o Instrucciones. - Orientados a la utilización por programadores profesionales. - Requiere especificación sobre cómo ejecutar una tarea. - Se deben especificar todas las alternativas. - Requiere gran número de instrucciones de procedimiento. - El código puede ser difícil de leer, entender y mantener. - Lenguaje creado originalmente para operación por lotes. - Puede ser difícil de aprender. - Difícil de depurar. - Orientados comúnmente a archivos.	• Fortran (FORmula TRANslation) • Basic (Beginner's AII-purpose Symbolic Instruction Code) • Cobol (COmmon Business-Oriented Language) • Pascal • Ada • ALGOL • Smalltalk • PL/l • C
ORIENTADA A OBJETOS La idea principal de la programación orientada a objetos es construir programas que utilizan objetos de software. Un objeto puede considerarse como una entidad independientemente de cómputo con sus propios datos y programación.	-Se basa en conceptos sencillos: objetos y atributos, el todo y las partes, clases y miembros. - Encapsulación - Herencia - Polimorfismo - Facilita la creación de prototipos - Simplicidad - Modularidad -Facilidad para hacer modificaciones - Posibilidad de extenderlo	• Java • PHP • ASP • PERL • BASH • Visual J ++
FUNCIONAL La programación funcional se basa en el uso de las funciones matemáticas para producir mejores efectos y que sus resultados sean más eficientes.	- Semánticas claras, simples y matemáticamente bien fundadas - Cercanos al nivel de abstracción de las especificaciones formales / informales de los problemas a resolver - Referencialmente transparentes: Comportamiento matemático adecuado que permite razonar sobre los programas -Soportan técnicas muy avanzadas de desarrollo, mantenimiento y validación de programas -Altas dosis de paralelismo implícito - Aplicaciones variadas y de gran interés - Provee de un paradigma para programar en paralelo. - Es ampliamente aplicada en la	• ML (MetaLanguage) • LISP

	Inteligencia Artificial. - Es bastante útil para el desarrollo de prototipos.	
LOGICA La programación lógica los programas se consideran como una serie de aserciones lógicas. De esta forma, el conocimiento se representa mediante reglas, tratándose de sistemas declarativos	- Emerge de la lógica de primer orden de la Lógica Matemática. - Busca la demostración automática de teoremas dentro de la Inteligencia Artificial, mediante un mecanismo de inferencia. - Se basa en el estudio de los lenguajes formales como parte de las ciencias de la computación. - Caracterización de propiedades y relaciones. - Doble dirección (E/S) de los datos. - Datos parcialmente construidos. -Teóricamente, el orden no Importa. -Sintaxis y semántica bien definidas -Reglas de inferencia	• PROLOG

6.3 PARADIGMA DEL ALGORITMO DE EUCLIDES.

Método básico:

En esta lección vamos a ver el Algoritmo de Euclides que es un método muy astuto para calcular el divisor común entre dos números. Una de sus principales ventajas es que no es necesario calcular los divisores ni los factores de los números, por lo que anda muy rápido, incluso cuando alguno de los números es un primo o tiene algún factor primo grande.

Este método se basa en la siguiente propiedad:

- **si A y B son enteros entonces DCM(A,B) = DCM(A-B,B)**

A continuación se puede ver el desarrollo.
La idea es ir achicando los números tanto como se pueda hasta que el DCM sea obvio, por ejemplo que sean ambos números iguales, o que uno de los dos sea cero.
Por ejemplo: para calcular el DCM(8069,5209)
DCM(8069,5209) =DCM(8069-5209,5209) =DCM(2860,5209) =DCM(5209-2860,2860) =DCM(2349,2860) =DCM(2860-2349,2349) =DCM(511,2349) =DCM(2349-511,511) =DCM(1838,511) =DCM(1838-511,511) =DCM(1327,511) =DCM(1327-511,511) =DCM(816,511) =DCM(816-511,511) =DCM(305,511) =DCM(511-305,305) =DCM(206,305) =DCM(305-206,206) =DCM(99,206) =DCM(206-99,99) =DCM(107,99) =DCM(107-99,99) =DCM(8,99) =DCM(99-8,8) =DCM(91,8) =DCM(91-8,8) =DCM(83,8) =DCM(83-8,8) =DCM(75,8) =DCM(75-8,8) =DCM(67,8) =DCM(67-8,8) =DCM(59,8) =DCM(59-8,8) =DCM(51,8) =DCM(51-8,8) =DCM(43,8)

=DCM(43-8,8) =DCM(35,8) =DCM(35-8,8) =DCM(27,8) =DCM(27-8,8) =DCM(19,8) =DCM(19-8,8) =DCM(11,8) =DCM(11-8,8) =DCM(3,8) =DCM(8-3,3) =DCM(5,3) =DCM(5-3,3) =DCM(2,3) =DCM(3-2,2) =DCM(1,2)= DCM(2-1,1) =DCM(1,1) =DCM(1-1,1) =DCM(0,1) =1

Si revisan con cuidado, van a ver que a pesar de que son muchas cuentas, todas son muy simples (solo hay que restar y restar y restar). En cambio si uno trata de calcular el divisor común mayor de la manera usual, puede pasarse bastante tiempo buscando cuales son los divisores de 8069 y 5209.

Veamos otro ejemplo: Calcular el DCM(1651,2353)

DCM(1651,2353) =DCM(702,1651) =DCM(949,702) =DCM(247,702) =DCM(455,247) =DCM(208,247) =DCM(208,247) =DCM(39,208) =DCM(169,39) =DCM(130,39) =DCM(91,39) =DCM(52,39) =DCM(13,39) =DCM(26,13) =DCM(13,13) =DCM(0,13) =13

Se puede hacer una función que haga estas cuentas para calcular el DCM entre dos números con el mismo formato que usábamos en las lecciones anteriores. (En realidad esta función anda bien si a y b son enteros positivos o cero.)

```
Deflng A-Z 'Todas las variables son enteros largos
Function DCMRestando(a,b)
   CopiaA=a 'Copia a A (ver nota mas abajo)
   CopiaB=b 'Copia a B (ver nota mas abajo)
   'Print "DCM(" ; CopiaA ; "," ; CopiaB ; ")"
   Do While CopiaB <> 0
      If CopiaA < CopiaB Then 'si están al revés los doy vuelta
         Temp = CopiaA
         CopiaA = CopiaB
         CopiaB = Temp
      End If
      CopiaA = CopiaA - CopiaB
      'Print "=DCM(" ; CopiaA ; "," ; CopiaB ; ")"
   Loop
   DCMRestando = CopiaA
   'Print "=" ; CopiaA
End Function
```

(Todas las salidas a pantalla están comentadas para que no molesten en un programa normal, pero puede ser útil activarlas para entender como funciona exactamente la función.)

Un poco más rápido:

Para acelerar un poco las cuentas se puede utilizar la división entera para hacer varios de estos pasos al mismo tiempo. Al dividir A por B el cociente representa la cantidad de veces que entra B completamente en A y el resto es lo que sobra para que la cuenta sea exacta, por lo que después de restarle B a A varias veces vamos a terminar obteniendo el resto.

Por ejemplo en el primer ejemplo aparece

DCM(2349,511) =DCM(1838,511) =DCM(1327,511) =DCM(816,511) =DCM(305,511)

y al hacer la división entera de 2349 dividido 511 se obtiene como resultado 4 y resto 305. Por esto se ve en el ejemplo hay que restarle 4 veces 511 y se obtiene como resultado 305.
De esta manera los ejemplos quedan:
DCM(8069,5209) =DCM(8069-5209,5209) =DCM(2860,5209) =DCM(5209-2860,2860) =DCM(2349,2860) =DCM(2860-2349,2349) =DCM(511,2349) =DCM(2349-4*511,511) =DCM(1838,511) =DCM(305,511) =DCM(511-305,305) =DCM(206,305) =DCM(305-206,206) =DCM(99,206) =DCM(206-2*99,99) =DCM(8,99) =DCM(99-12*8,8) =DCM(3,8) =DCM(8-2*3,3) =DCM(2,3) =DCM(3-2,2) =DCM(1,2)= DCM(2-2*1,1) =DCM(0,1) =1
DCM(1651,2353) =DCM(702,1651) =DCM(949,702) =DCM(247,702) =DCM(208,247) =DCM(208,247) =DCM(39,208) =DCM(13,39) =DCM(0,13) =13
Con esta observación puede mejorar un poco la función de la siguiente manera usando el resto de la división entera. En esta función no se necesita verificar si el valor de a es mayor que el de b, porque si a fuera más chico la primera división de 0 y sólo se dan vuelta los números.

```
Deflng A-Z 'Todas las variables son enteros largos
Function DCMEuclides(a,b)
   CopiaA=a 'Copia a A (ver nota mas abajo)
   CopiaB=b 'Copia a B (ver nota mas abajo)
   Do While CopiaB<>0'Repite hasta que el resto de cero
      Temp = CopiaA Mod CopiaB
      'Como el resto de la división es siempre menor que
      'CopiaB entonces los doy vuelta
      CopiaA = CopiaB
      CopiaB = Temp
   Loop
   DCMEuclides = CopiaA
End Function
```

Es sorprendente lo corta que es esta función y además anda realmente muy rápido, así que en (¿casi?) todos los casos es preferible usar esta función a todas las que vimos antes.
Nota: En ambos programas se hacen copias de los parámetros a y b para operar con las copias y no modificarlos. Esto es necesario porque sino al cambiarse los valores dentro de la función, se cambian los valores de las variables en el programa principal (o de donde se llame a la función). Esto sólo es necesario en QB y VB, pero no en Pascal, C, etc. porque en estos lenguajes la función recibe normalmente sólo el valor de las variables. En general uno ni se preocupa por todo esto, hasta que el programa comienza a hacer cosas inexplicables.

Comentarios:

Cuando hay que hacer este tipo de cuentas a mano, a veces se pueden utilizar otras propiedades, como por ejemplo, es obvio cuando un número es múltiplo de 10, o cuando es par. Así que también se podrían resolver los ejemplo anteriores de la siguiente manera:
DCM(8069,5209) =DCM(2860,5209) =DCM(286*10,5209) =DCM(286,5209) =DCM(143*2,5209) =DCM(143,5209) =DCM(143,5209) =DCM(5209-143*3,143) =DCM(4780,143) =DCM(239*20,143) =DCM(239,143) =DCM(96,143) =DCM(49,96) =DCM(7*7,3*31) =1

DCM(1651,2353) =DCM(702,1651) =DCM(351,1651)
=DCM(1300,351)=DCM(13,351)=13
Sin embargo, la mayoría de estos trucos que se pueden hacer a ojo usando las reglas de divisibilidad más sencillas no son tan útiles para utilizar en un programa.

6.4 LOS PARADIGMAS NUMERICOS.

Los algoritmos se clasifican en numéricos y los no numéricos.

Los algoritmos numéricos emplean la definición de los datos, el tipo de dato, variables y constantes; la utilización de procedimientos y funciones; los parámetros; las etiquetas; las instrucciones, las declaraciones y sentencias; en casos recientes los objetos y los eventos.

Todo lo anterior mencionado se anoto como si fuese estático, pero no lo es, al momento de la codificación, cada algoritmo toma personalidad y el estilo dl programador le va dando el verdadero alcance de acuerdo a la aplicación especifica; no obstante los estios de programacon como lo son: el secuencial, el modular, el orientado a eventos le van dando a la construcción de las instrucciones las características specilaes en la definición de ls instrucciones. Afortunadamente los valore se van almacenando en variables las cuales se pueden cambiar de acuerdo a otros parametros

6.4.1 LA INTERPOLACIÓN.

El siguiente algoritmo es la página del Instituto Universitaria Politécnica.

Consta de las siguientes partes:

• Introducción a la teoría de interpolación. Interpolación polinomial, casos particulares.
• Polinomio de interpolación de Lagrange.
• Construcción del polinomio de interpolación de Lagrange: Fórmula de Lagrange.
• Construcción del polinomio de interpolación de Lagrange: Fórmula de Newton.
• Polinomio de interpolación de Lagrange: Cotas de error.
• Algoritmo de Aitken.

1.- INTRODUCCIÓN A LA TEORÍA DE INTERPOLACIÓN.
INTERPOLACIÓN POLINOMIAL: CASOS PARTICULARES.
Un problema clásico de la matemática, utilizado con frecuencia en la práctica, es el de calcular
el valor de una función dada en un punto cuando no se puede o no se desea, por las razones que
sean, evaluar directamente la expresión de la función. Supóngase que, por ejemplo, se han efectuado
experiencias que permiten conocer una función para determinados valores de su variable y se desea

conocer el valor de la función en puntos intermedios entre aquellos.
La resolución aproximada del problema consiste en encontrar una función fácil de construir y,
a la vez, fácil de evaluar y que coincida con la función objeto del problema en algunos datos que
conocemos sobre ésta. Entonces se dice que la función así construida interpola a la función respecto
a los datos.
Ahora bien, aquí hay que concretar dos cosas principalmente:
1.- Los datos que han de ser comunes a la función dada y a la que la va a interpolar.
2.- Qué tipo de función se va a usar como función interpoladora o función de interpolación.
El ejemplo más típico –que va a ser el objetivo central de este tema– es el siguiente: sea f(x)
una función cuyo valor se conoce en N + 1 puntos $x_0, x_1, ..., x_N$ y deseamos encontrar su valor
aproximado para un valor x cualquiera. Para ello intentaremos encontrar un polinomio, de grado no
mayor que N, que tome los valores $f(x_k)$ en los puntos x_k para k = 0, 1, . . . , N. Con esto último,
hemos concretado los datos comunes y qué función vamos a usar como interpoladora: un polinomio de
grado no mayor que N. Este ejemplo da lugar a lo que se conoce como problema de interpolación
Polinomial clásico.
En la situación anterior, la función interpoladora que se quiere utilizar es un polinomio. Sin
embargo, podríamos proponer, por ejemplo, que la función de interpolación fuese una función del
tipo
$g(x) = c_0Ö_0(x) + c_1Ö_1(x) + · · · + c_NÖ_N(x)$.
Es decir, que la función de interpolación sea una combinación lineal de unas funciones dadas $Ö_0$,
$Ö_1, ..., Ö_N$, que verifique que g(xk) = (xk), k = 0, 1, . . . , N. De forma análoga, se podrían haber
impuesto otras condiciones a los datos que deben ser comunes a la función f(x) y a la función
Interpoladora g(x).
Así, además del problema de interpolación polinomial clásico, existen otros muchos casos particulares e interesantes de interpolación. De entre ellos, y dentro de los que utilizan los polinomios
como funciones de interpolación, son usuales los siguientes
a) Conocido el valor de una función f(x) y de sus N primeras derivadas sucesivas en un determinado
punto x0, encontrar un polinomio P(x), de grado menor o igual que N, tal que
P(x0) = f(x0), $P^{(k)}$**$(x_0) = f^{(k)}(x_0)$, k= 1, 2, . . . N.**

Este planteamiento se denomina problema de interpolación de Taylor.
b) Conocido el valor de una función f(x) y de su primera derivada $f_0(x)$, buscar un polinomio
P(x), de grado menor o igual que 2N + 1, tal que coincida él y su primera derivada con los
valores de la función f(x) y $f_0(x)$ en unos puntos $x_0, x_1, ..., x_N$. Es decir,
$P(x_k) = f(x_k)$, $P_0(x_k) = f_0(x_k)$, k= 0, 1, . . . N.
Este planteamiento es conocido por problema de interpolación de Hermite.
Lógicamente, en todas las situaciones que se planteen, especificada el tipo de función de interpolación
que se va a utilizar y establecidas las condiciones que han de ser comunes a la función dada

y a la que la va a interpolar, surgen, de manera inmediata, dos cuestiones básicas que tendrán que
contestarse: ¿Existe tal función de interpolación que verifica las condiciones impuestas? En caso de
existir, ¿es única?

2.- POLINOMIO DE INTERPOLACIÓN DE LAGRANGE.

Definición.
Sean N+1 puntos $x_0, x_1, . . . , x_N$, distintos dos a dos, y los correspondientes valores de una función
$y_0 = f(x_0), y_1 = f(x_1), . . . , y_N = f(x_N)$. Se denomina polinomio de interpolación de Lagrange
a un polinomio $P_N(x)$, de grado menor o igual a N, tal que
$P_N(x_0) = y_0, P_N(x_1) = y_1, ... , P_N(x_N) = y_N$.
Los valores x_i (i = 0, . . . , N) se denominan abscisas o nodos de la interpolación.
Teorema.
Dados N + 1 puntos $x_0, x_1, . . . , x_N$, distintos dos a dos, y los correspondientes valores de una
función $y_0 = f(x_0), y_1 = f(x_1), . . . , y_N = f(x_N)$, existe un único polinomio de interpolación de
Lagrange.
El procedimiento de cálculo directo del polinomio de interpolación de Lagrange (establecido en la
Demostración del teorema anterior) es laborioso. Buscamos otras formas de calcular más fácilmente
el polinomio de interpolación de Lagrange.

3.- CONSTRUCCIÓN DEL POLINOMIO DE INTERPOLACIÓN DE LAGRANGE: FÓRMULA DE LAGRANGE.

Para construir el polinomio $P_N(x)$, de grado menor o igual que N, que pase por los puntos
$(x_0, y_0), . . . , (x_N, y_N)$, puede utilizarse la fórmula

$$P_N(x) = \sum_{k=0}^{N} y_k L_{N,k}(x), \quad (1)$$

siendo $L_{N,k}(x)$ =
$(x - x_0) \ldots (x - x_{k-1})(x - x_{k+1}) \ldots (x - x_N)$
$(x_k - x_0) \ldots (x_k - x_{k-1})(x_k - x_{k+1}) \ldots (x_k - x_N)$
.
Es decir, $P_N(x)$ se expresa como una combinación lineal de N + 1 polinomios de grado N.
Un cálculo directo prueba que, para cada k fijo,
$L_{N,k}(x_i)$ = (1 si i = k
0 si i 6= k
3
con lo que $P_N(x_i)$ =
$\sum_{k=0}^{N}$
$y_k L_{N,k}(x_i) = y_i$ y, puesto que el polinomio de interpolación de Lagrange es Único, la fórmula (1) nos proporciona un procedimiento válido de cálculo. A esta forma de expresión
De $P_N(x)$ se la denomina fórmula de Lagrange del polinomio de interpolación.
4.- CONSTRUCCIÓN DEL POLINOMIO DE INTERPOLACIÓN DE LAGRANGE: FÓRMULA DE NEWTON.
Los procedimientos anteriores para la construcción del polinomio interpolador de Lagrange tienen
el inconveniente de que no existe relación entre la construcción de $P_{(N-1)}(x)$ y $P_N(x)$. Ahora, vamos
a seguir un procedimiento de construcción distinto debido a Newton y que tiene como idea básica la
de construir la solución en pasos sucesivos.
Consideremos N + 1 puntos $x_0, x_1, \ldots, x_N$, distintos dos a dos, y los correspondientes valores de
una función $y_0 = f(x_0), y_1 = f(x_1), \ldots, y_N = f(x_N)$ para los que se quiere determinar el polinomio
de interpolación de Lagrange. Se va a proceder de la siguiente forma: Primero, se construye el
polinomio $P_0(x)$, de grado menor o igual que 0 (es decir, constante) que coincida con la función en
el punto x_0. Luego, se construye el polinomio $P_1(x)$, de grado menor o igual que 1, que coincida con
la función en x_0, x_1. A continuación, el polinomio $P_2(x)$, de grado menor o igual que 2, que coincida
con la función en x_0, x_1, x_2, y así hasta llegar al polinomio $P_N(x)$, de grado menor o igual que N,
que satisfaga todas las condiciones establecidas.
El procedimiento descrito en el párrafo anterior da lugar al siguiente esquema recursivo:
$P_0(x) = c_0$,
$P_1(x) = c_0 + c_1(x - x_0)$,
$P_2(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1)$,
$P_3(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1)$

$+c_3(x - x_0)(x - x_1)(x - x_2)$,
...
$P_N(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1)$
$+c_3(x - x_0)(x - x_1)(x - x_2)$
$+c_4(x - x_0)(x - x_1)(x - x_2)(x - x_3) + \cdots$
$+c_N(x - x_0)(x - x_1)(x - x_2)(x - x_3) \cdots (x - x_{N-1})$,
a lo largo del cual se irían imponiendo las condiciones de interpolación
$P_0(x_0) = f(x_0)$, $P_1(x_1) =$
$f(x_1)$, ..., $P_N(x_N) = f(x_N)$, para calcular sucesivamente los números $c_0, c_1, \ldots, c_N$.
Obsérvese que, entonces, el polinomio $P_N(x)$ se obtiene a partir de $P_{N-1}(x)$ usando la recurrencia
$P_N(x) = P_{N-1}(x) + c_N(x - x_0)(x - x_1)(x - x_2)(x - x_3) \cdots (x - x_{N-1})$,

y queda ahora escrito como una combinación lineal de
$(x - x_0), (x - x_0)(x - x_1), \ldots, (x - x_0)(x - x_1) \cdots (x - x_{N-1})$.
A esta forma de expresión de $P_N(x)$, esto es
$P_N(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots + c_N(x - x_0)(x - x_1) \cdots (x - x_{N-1})$
se la denomina fórmula de Newton del polinomio de interpolación de Lagrange.
Ahora, vamos a dar un medio de calcular las constantes c_i que aparecen en el proceso recursivo de
construcción del polinomio de interpolación de Lagrange. Para ello, introducimos, en primer lugar,
una definición.
Definición.
Se llama diferencia dividida de una función f(x) en un punto z_i, y escribimos $f[z_i]$, al valor de
la función f(x) en el punto z_i. Es decir,
$f[z_i] = f(z_i)$.
La diferencia dividida de f(x) en dos o más puntos distintos dos a dos, $z_1, z_2, \ldots, z_k$, se denota
por $f[z_1, z_2, \ldots, z_k]$ y se define como

$$f[z_1, z_2, \ldots, z_k] = \frac{f[z_2, \ldots, z_k] - f[z_1, \ldots, z_{k-1}]}{z_k - z_1}$$

.
Teorema.
Dados N + 1 puntos $x_0, x_1, \ldots, x_N$, distintos dos a dos, y los correspondientes valores de una
función $y_0 = f(x_0), y_1 = f(x_1), \ldots, y_N = f(x_N)$, la fórmula de Newton del único polinomio de
interpolación de Lagrange viene dada por
$P_N(x) = c_0 + c_1(x - x_0) + \cdots + c_N(x - x_0)(x - x_1) \cdots (x - x_{N-1})$,
siendo $c_k = f[x_0, x_1, \ldots, x_k]$ para $k = 0, 1, \ldots, N$.
Teniendo en cuenta el teorema anterior, los coeficientes c_k, que permiten construir el polinomio

interpolador de Lagrange siguiendo la fórmula de Newton, son diferencias divididas de la función f(x)
las cuales pueden ser fácilmente calculadas utilizando la definición recurrente dada y disponiendo los
Cálculos como se indican en la siguiente tabla para el caso N = 4.
5
x_k $f[x_k]$ $f[\ ,\ ]$ $f[\ ,\ ,\ ]$ $f[\ ,\ ,\ ,\ ]$ $f[\ ,\ ,\ ,\ ,\ ]$
x_0 $f[x_0]$
x_1 $f[x_1]$ $f[x_0, x_1]$
x_2 $f[x_2]$ $f[x_1, x_2]$ $f[x_0, x_1, x_2]$
x_3 $f[x_3]$ $f[x_2, x_3]$ $f[x_1, x_2, x_3]$ $f[x_0, x_1, x_2, x_3]$
x_4 $f[x_4]$ $f[x_3, x_4]$ $f[x_2, x_3, x_4]$ $f[x_1, x_2, x_3, x_4]$ $f[x_0, x_1, x_2, x_3, x_4]$
Se observa que, en general, construida la tabla correspondiente a los datos $(x_0, y_0), \ldots, (x_N, y_N)$,
los elementos diagonales nos dan los coeficientes por los que hay que multiplicar los polinomios
$(x - x_0), (x - x_0)(x - x_1), \ldots, (x - x_0)(x - x_1) \cdot \cdot \cdot (x - x_{N-1})$,
para formar el polinomio de interpolación de Lagrange en su forma de Newton.

5.- POLINOMIO DE INTERPOLACIÓN DE LAGRANGE: COTAS DE ERROR.

Formado, por el procedimiento que sea, el polinomio de interpolación de Lagrange, surge la
pregunta: ¿Qué diferencia hay entre su valor P(x) en un punto x dado y el verdadero valor f(x) de
la función?
Teorema.
Sea f(x) una función con N = 1 derivadas continuas en un intervalo [a, b] y tal que $f^{(N+1)}(x)$
existe en (a, b). Sean $x_0, x_1, \ldots, x_N$, N + 1 nodos en [a, b], distintos dos a dos, y sea $P_N(x)$ el
correspondiente polinomio interpolador de Lagrange. Entonces, para cada punto x en [a, b],

$$E_N(x) = f(x) - P_N(x) = \frac{(x - x_0)(x - x_1) \cdot \cdot \cdot (x - x_N)}{(N + 1)!} f^{(N+1)}(\hat{\imath})$$

para algún valor $\hat{\imath} = \hat{\imath}(x)$ del intervalo (a, b).
Si, a las hipótesis del teorema anterior, se le añade que, para todo x . [a, b], $|f^{(N+1)}(x)| = K_{N+1}$
entonces

$$|E_N(x)| = |f(x) - P_N(x)| = \frac{|x - x_0||x - x_1| \cdot \cdot \cdot |x - x_N|}{(N + 1)!} K_{N+1}, \ .x \ . \ [a, b].$$

Además, en el caso especial de que los N + 1 nodos estén equiespaciados pueden darse resultados
más específicos para la cota del error. Es decir, sí
$a = x_0 < x_1 < \cdot \cdot \cdot < x_N = b$, siendo $x_k = x_0 + hk$, para $k = 0, 1, \ldots, N$,

6
entonces, los términos del error correspondientes a los casos N = 1, N = 2 y N = 3 admiten las
siguientes cotas
$|E_1x)| = |f(x) - P_1(x)| =$
h_2
8
K_2, válida para x . $[x_0, x_1]$,
$|E_2x)| = |f(x) - P_2(x)| =$
h_3
9v3
K_3, válida para x . $[x_0, x_2]$,
$|E_3x)| = |f(x) - P_3(x)| =$
h_4
24
K_4, válida para x . $[x_0, x_3]$.
Por otro lado, también es conveniente plantearse si el error de interpolación $f(x) - P_N(x)$ puede
Hacerse arbitrariamente pequeño incrementando el grado del polinomio. Más precisamente: dada
una función f(x) definida en un intervalo [a, b], elijamos un punto $x^{(0)}_0$ e interpolemos en él por un
polinomio constante $P_0(x)$; elijamos dos puntos distintos entre sí $x^{(1)}_0$, $x^{(1)}_1$ e interpolemos en ellos
por una recta $P_1(x)$; elijamos tres puntos, distintos dos a dos, $x^{(2)}_0$, $x^{(2)}_1$, $x^{(2)}_2$ e interpolemos por
una parábola cuadrática, etc. ¿Será cierto que $\lim_{N.8} P_N(x) = f(x)$? La respuesta es, en general,
negativa, con la conclusión de que incrementar el grado de los polinomios de interpolación no siempre
es aconsejable.

6.- POLINOMIO DE INTERPOLACIÓN DE LAGRANGE: ALGORITMO DE AITKEN.

En esta sección vamos a dar un procedimiento recurrente para calcular el polinomio de interpolación
de Lagrange para el conjunto de datos $\{(x_k, y_k\}_{k=0}^N$. Se basa en un resultado que se expondrá
a continuación.
Previamente, hay que indicar que si consideramos el conjunto de datos $\{(x_k, y_k)\}_{k=0}^N$, el conjunto
de nodos $A = \{x_0, x_1, \ldots, x_N\}$ y un subconjunto no vacío B de A, denotaremos por $P_B(x)$ el polinomio

de intepolación de Lagrange correspondiente a los nodos que forman el conjunto B.

Teorema. (Lema de Aitken)

Sean B y C dos subcojuntos no vacíos de A, con todos sus nodos comunes excepto x_j. B que no está en C y x_i. C que no está en B. Entonces,

$$P_{B.C}(x) = \frac{(x_i - x)P_B(x) - (x_j - x)P_C(x)}{x_i - x_j}$$

.

Del teorema se desprende que es posible construir el polinomio que interpola a una función en N+1

7

nodos utilizando dos polinomios que la interpolan en N nodos. Por ejemplo, para conocer el polinomio de interpolación correspondiente a los nodos $\{x_0, x_1, x_2, x_3\}$ podemos utilizar los polinomios asociados a $\{x_0, x_1, x_2\}$, $\{x_0, x_1, x_3\}$, y para éstos los de $\{x_0, x_1\}$, $\{x_0, x_2\}$ y $\{x_0, x_1\}$, $\{x_0, x_3\}$, respectivamente.

Por ello, es conveniente disponer los cálculos como aparece en la siguiente tabla

x_0 x_0 -x y_0

x_1 x_1 -x y_1 P{x0,x1}(x)

x2 x2 -x y2 P{x0,x2}(x) P{x0,x1,x2}(x)

x3 x3 -x y3 P{x0,x3}(x) P{x0,x1,x3}(x) P{x0,x1,x2,x3}(x)

Obsérvese, en la tabla, que a partir de y_0 y de cada uno de los y_1, y_2, y_3 se construye, mediante la fórmula que nos proporciona el lema de Aitken, la columna de los $P_{\{x_0,x_i\}}(x)$. Después, a partir de $P_{\{x_0,x_1\}}(x)$ y de cada uno de los elementos de su columna se construye la siguiente columna, etc.

Para esto viene bien poner una columna previa con los $x_k - x$ ya que para obtener, por ejemplo, $P_{\{x_0,x_1,x_3\}}(x)$ necesitamos manejar $P_{\{x_0,x_1\}}(x)$, $x_1 - x$, $P_{\{x_0,x_3\}}(x)$ y $x_3 - x$.

Si se desea calcular únicamente el valor concreto de $P_{\{x_0,x_1,x_2,x_3\}}(x)$ en un punto x = c, basta sustituir x por c en todos los cálculos hechos en la tabla anterior. Al final se obtendrá $P_{\{x_0,x_1,x_2,x_3\}}(c)$; es decir, el valor deseado.

Por último hay que indicar que, lógicamente, la elección que se ha hecho de B y C para aplicar el lema de Aitken no es única. La que aquí se ha elegido es la que da lugar al algoritmo de Aitken:

$$P_{\{x_0,x_1,\ldots,x_{i-1},x_i\}}(x) = \frac{(x_i - x)P_{\{x_0,x_1,\ldots,x_{i-1}\}}(x) - (x_{i-1} - x)P_{\{x_0,x_1,\ldots,x_{i-2},x_i\}}(x)}{x_i - x_{i-1}}$$

.
8

6.4.2 CALCULOS MATRICIALES.

Algoritmo utilizando los Indices de Welch

Una aproximación tentativa en tratar de utilizar los resultados de trabajos de autores anteriores como los del Dr. Harold V. McIntosh y Jose Manuel Gómez Soto; este algoritmo consiste en aprovechar los conceptos de multiplicidad uniforme e índices de Welch para la construcción de las posibles matrices de evolución de ACLR; por ejemplo, para un ACLR(5,h) se tiene la siguiente ``plantilla'' para generar las matrices de evolución:

	0	1	2	3	4
0	0				
1		1			
2			2		
3				3	
4	4	4	4	4	4

Tabla: **Plantilla para generar matrices de evolución de ACLR(5,h).**
El resto de los lugares vacíos se llenaban con todas permutaciones de estados posibles de tal manera que un elemento de cada columna fuera diferente al resto, esto es ya que al tener cinco nodos el diagrama de de Bruijn asociado, el valor de LMR=5, por lo que se fija a M=1 y L=1, lo que implica que no debe existir dos elementos iguales por columna.
La razón por la cual el último renglón de la matriz estaba lleno del estado 4 era para tener un estado en el cual fuera seguro que el conjunto de todas sus extensiones compatibles derechas tuviera una cardinalidad igual a R, cumpliendo para ese estado que LMR=5, una vez llena la matriz, se generaba una evolución de este ACL y se observaba si cada posible vecindad de longitud 2 tenía un único estado para el cual evolucionar hacia atrás, de cumplir ésto, el ACL se tomaba como reversible, de no ser así se verificaban si las vecindades de longitud 3 si cumplían con tener un único estado en el pasado; este proceso continuaba hasta revisar vecindades de longitud 4.

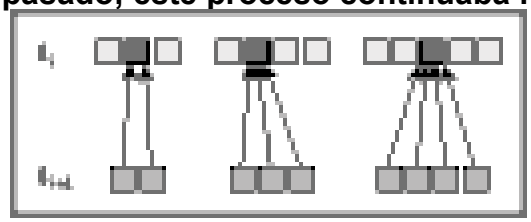


Figura: **Verificación de que cada cadena tenga un único elemento en común en el pasado para cadenas de longitud 2, 3 y 4.**
La ventaja de este planteamiento es que ya no hace la revisión de cada posible regla de evolución para un ACL(k,h) sino que solo analiza aquellas matrices de evolución en donde se tenga una diagonal principal donde cada estado sea diferente al resto y un renglón formado por un mismo valor para asegurar que la cardinalidad del conjunto de extensiones compatibles por la

derecha del mismo fuera R, con lo que el tiempo de computación que toma hacer el cálculo se reduce de manera significativa.

BIBLIOGRAFÍA.

Diccionario El Pequeño Larousse Ilustrado, Edit. Larousse, edición 1999.