LA TECNICA DE LA SIMULACION DIGITAL EN LA INVESTIGACION DE OPERACIONES

tesis para obtener el grado de maestro

en Investigación de Operaciones

presentada por el alumno:

José Maclovio Sautto Vallejo

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

La ciencia no tiene fronteras, porque el conocimiento es patrimonio de toda la humanidad, es la antorcha que ilumina el mundo. La ciencia es la más alta realización de un país, porque la nación impulsa el progreso del pensamiento y la inteligencia.

Louis Pasteur, químico francés, microbiólogo (1822-1895).

RESUMEN.

El desarrollo de la computación, ha posibilitado una amplia divulgación de técnicas de la Investigación de Operaciones, para la solución de diversos problemas. Así vemos, que en algunas hojas de cálculo, entre las diversas posibilidades que presentan se encuentra la solución de modelos de programación lineal.

Una de las técnicas, que más se ha beneficiado con el desarrollo de la computación es la simulación digital. Un rasgo que diferencia a esta técnica, de otras, es que permite el modelado de sistemas complejos, es aquí donde el enfoque sistémico tiene amplias posibilidades de concreción en su modelado e interpretación de resultados.

Este trabajo, presenta una síntesis del desarrollo de la simulación digital y plantea que el progreso de los medios de comunicación, nos permiten el acceso a los conocimiento más recientes, así como, a los últimos adelantos en las herramientas de programación y de equipos de computo, todo ello a costos relativamente bajos. Estos hechos, nos ponen por primera vez en la historia reciente, en igualdad de circunstancias para desarrollar esta técnica. Nos deja ver, que el problema de crecimiento en la simulación, es un problema teórico-practico al que hay que enfrentar resueltamente.

El principal obstáculo al que tenemos que vencer para enfrentar este reto, son las metas propias del subdesarrollo que con frecuencia gravitan sobre nuestras cabezas -producto de nuestro herencia colonial-, desterrar estos planteamientos y asumir los retos de nuestro tiempo es nuestra tarea.

El presente trabajo consta de cuatro capítulos y un anexo. En el primer capítulo se proporciona una descripción de lo que son los modelos y su relación con la computación para lo cual se da una breve introducción histórica del desarrollo de la computación y su impacto en la Investigación de Operaciones, y se discuten características generales de la modelación de sistemas, particularizando en las diferencias de los modelos analíticos y los heurísticos. En el capítulo dos se presenta una síntesis del desarrollo de la simulación digital y se plantean posibles lineas de desarrollo. En el capítulo tres se desarrolla un estudio de caso de una institución bancaria donde se aplica una de las alternativas planteadas en el capitulo dos, este capítulo consta de dos apartados que corresponden al manual de usuario del sistema desarrollado y el manual técnico del mismo. Finalmente se desarrolla un último apartado de conclusiones que se derivan de este trabajo.

Se incluyen también un anexo, sobre los conceptos probabilísticos más utilizados en simulación, que incluye procedimientos en pascal, de las funciones de distribución más frecuentes en la generación de números aleatorios, tales como la normal, exponencial y uniformes en un intervalo, en este último caso, para variables discretas y continuas. Y también se presentan algunos conceptos básicos de teoría de colas.

Es el acontecimiento central, la clave de la comprensión de los años inmediatamente venideros. Es un acontecimiento tan profundo como aquella primera ola de cambio desencadenada hace diez mil años por la invención de la agricultura, o la sísmica segunda ola de cambio disparada por la revolución industrial. Nosotros somos hijos de la transformación siguiente, la tercera ola.

Tratamos de encontrar palabras para describir toda la fuerza y el alcance de este extraordinario cambio. Algunos hablan de una emergente era espacial, Era de la información, Era electrónica o Aldea global. Zbigniew Brzezinski nos ha dicho que nos hallamos ante una " era tecnetrónica ". El sociólogo Daniel Bell describe el advenimiento de una " sociedad postindustrial ". Los futuristas soviéticos hablan de la RCT, la " revolución científico-tecnológica ". Yo mismo he escrito extensamente sobre el advenimiento de una "sociedad superindustrial ". Pero ninguno de estos términos incluido el mío, es adecuado.

Alvin Toffler "La tercera ola".

INTRODUCCION

La Investigación de Operaciones (I. de O.), forma parte de un conjunto de nuevas ciencias, producto directo de la revolución científico tecnológica, cuya oleada de transformación ha venido y continua trastocando nuestras formas de vida. A partir de mediados del presente siglo, hemos sido testigos de un impresionante cambio sustentado en la electrónica, más concretamente en la computación.

Hace treinta años, a principios de los 60's, cuando las primeras grandes computadoras, irrumpieron en las industrias, ministerios y universidades se pensó que esta nueva tecnología, que ese gran "megacerebro" electrónico sería capaz de centralizar y controlar toda la información necesaria para conducir una institución, asi se crearon una serie de mitos alrededor de las computadoras como "solo los superdotados pueden manipularlas", "solo manejan ceros y unos y es difícil interpretar los resultados", etc. La verdad era, que durante esa época, más que una superinteligencia se requería una paciencia de monje tibetano para programar, corregir y leer los resultados obtenidos, y sobre todo, obtener tiempo de máquina.

Una década después, en los años 70, los microprocesadores llegaron, los tiempos de jalarse los cabellos, arrastrar los pies y de ir a mendigar unos pocos minutos de tiempo de computadora, llegaba a su fin. La revolución de las micros dio al traste con el monopolio de la información, los costos por el uso de computadoras se fue

a tierra, y sobre todo desmistificó a la computación misma; ahora la computadora era accesible a todos y cada día aparecían nuevos programas que volverían más accesible el uso de los ordenadores, el microprocesador fue el sepulturero de la idea "solo los superdotados usan las computadoras".

En los últimos 10 años, palabras como software y hardware, se han convertido de uso corriente en nuestras universidades, empresas e instituciones públicas. El desarrollo tecnológico permite que el acceso a la información sobre el desarrollo científico y tecnológico sea más expedito, en la actualidad es posible recibir publicaciones científicas y de divulgación, editadas en cualquier parte del mundo, el mismo mes en que se publican.

El desarrollo del software especializado descansa sobre dos pilares, el primero constituido por herramientas de la computación tales como los lenguajes y los procedimientos de apoyo, y el segundo por los conocimientos, habilidad e ingenio de nuestros diseñadores y programadores.

El acceso a las herramientas de programación, está al alcance de todos, de donde el desarrollo de software especializado es un problema práctico, donde nuestra pertenencia a un país tercer mundista no es pretexto para enfrentar este nuevo reto de nuestro tiempo.

Por primera vez, podemos plantearnos competir con cualquiera, en el desarrollo de software, que resuelva problemáticas de las diversas empresas y/o instituciones públicas.

En este contexto, me permito presentar mis reflexiones sobre el desarrollo de la simulación en el análisis de sistemas, planteando como opciones de evolución:

- La elaboración de modelos iterativos, con un usuario neófito en programación y
- El desarrollo de modelos de simulación de grandes instituciones mediante redes.

CAPITULO I

FUNDAMENTOS DEL MODELADO

"...todo retrato pintado con emoción es un retrato del artista, no del modelo. Este no es más que el accidente, la ocasión. No es él el revelado por el pintor, sino mas bien éste quien, sobre el lienzo pintado, se revela a sí mismo".

Oscar Wilde: "El retrato de Dorian Gray"

La simulación es una de las herramientas de análisis mas poderosas de que disponen quiénes se dedican al diseño y operación de procesos o sistemas complejos. El concepto de simulación es simple e intuitivamente atractivo, le permite al usuario experimentar con sistemas reales o ficticios en casos en los que de otra manera esto sería imposible o impráctico.

El modelado de simulaciones se basa principalmente en la computación, la matemática, la probabilidad y la estadística, es por esto que el propósito de éste primer capítulo es presentar las bases de la modelación en general y en particular la modelación de sistemas, así como la forma en que se puede utilizar la computadora para éste propósito; se presenta en la primera sección un breve desarrollo histórico de la computadora así como una tabla donde se muestran los avances mas significativos de las computadoras en los últimos veinte años. En la segunda sección se define un modelo y se describe la función de los modelos. En la tercera se da una clasificación de modelos analíticos y heurísticos y se describe cada uno de ellos, finalmente en la cuarta sección se desarrolla lo que es la modelación de sistemas.

1.1 LA COMPUTADORA.

Así como la máquina de vapor sentó las bases materiales para el inicio de la Revolución Industrial, la computadora, la niña consentida de la electrónica, sentó las bases materiales para esta nueva revolución, que en pocos años ha transformado nuestras formas de vida, poco a poco sin darnos cuenta, el cambio se ha convertido en la única constante de nuestra forma de vida.

Esta nueva revolución, ha impactado todas las esferas de la sociedad. Dentro del conocimiento humano, ha facilitado el desarrollo de las viejas áreas del conocimiento y creado nuevas ciencias, tales como la informática, la cibernética y desde luego la Investigación de Operaciones. Basta dar un simple repaso a los hechos históricos más relevantes durante los últimos 40 años.

En 1946, se construye la primera computadora (ENIAC), y con ella la posibilidad real de poder realizar cálculos aritméticos miles o tal vez millones de veces más rápido que los seres humanos. Este hecho, permitió la concretización de métodos de solución iterativos que involucran un gran número de operaciones aritméticas. Un año después en el verano de 1947 George Dantzig desarrolló completamente el método simplex para resolver problemas de programación lineal. Sin embargo fue hasta enero de 1952 cuando se realizó la primera solución exitosa de un problema de programación lineal en una computadora electrónica de "alta velocidad", en la maquina SEAC del National Bureaus of Standards. En la década de los 50's, se adelantó la teoría necesaria y se establecieron algoritmos que permitían resolver problemas de programación lineal, programación dinámica, líneas de espera, inventarios, etc.

En sus primeros 20 años, la Investigación de Operaciones (I. de O.) elaboró la teoría para enfrentar exitosamente, la mayoría de las problemáticas que se generan en la pequeña y mediana industria. Y para mediados de los años 70's, ocupaba ya un lugar importante en la Industria y Universidades, popularizándose el estudio de sus técnicas en los países desarrollados. En estos mismos 20 años la electrónica a partir del invento del transistor en 1948 y de los circuitos integrados en 1959 permitió el desarrollo de nuevas y mejores computadoras que año con año se mejoran, en rapidéz, precisión y en el manejo de grandes volúmenes de información.

Los 70's, vieron nacer una nueva generación de computadoras, las micros, que en pocos años, han alcanzado dimensiones verdaderamente impresionantes, en rapidez, en volúmenes de información que manejan, y sobre todo en abaratamiento de sus costos. Las microcomputadoras que hoy encontramos en cualquier Universidad, son superiores a las primeras grandes computadoras que se desarrollaron

durante los años 50's, lo que ha permitido que la I. de O. las utilice con mayor frecuencia y desarrolle la simulación digital para realizar modelos cada vez más apegados a la realidad, cuya modelación matemática no es posible.

La tabla 1.1 nos da una idea de la rápida evolución de la industria de las microcomputadoras.

Durante la década de los 80, se empezó a hablar de la programación en paralelo y de la utilización de redes de computadoras, la programación en paralelo supone la existencia de nuevas arquitecturas de computadoras, con varios procesadores. Mientras que la utilización de redes de microcomputadoras posibilitan el uso de las viejas microcomputadoras. Cualquiera que sea el desarrollo de las computadoras, en la década de los 90's tendrá la posibilidad de programar en paralelo o simular dicha programación mediante redes de computadoras abriendo las puertas a nuevas épocas en el desarrollo de las técnicas de la I. de O. sobre todo de la simulación digital.

TABLA 1.1

AÑO	ACONTECIMIENTO
1948	John Bardeen, Walter Brattain y William Shockley de los laboratorios Bell inventan el transistor.
1959	La Texas Instruments descubre el primer circuito integrado.
1964	John G. Kemeny y Thomas E. Kurtz desarrollan el lenguaje de programación BASIC en el Dartmouth College. La Digital Equipment Corp. anuncia la minicomputadora PDP-8 con un costo de $16, 200.
1970	Scientific American publica en la sección de Martin Gardner "Recreaciones Matemáticas" el juego de la vida de John Conway.
1971	La Intel Corp. pone el microprocesador de 4-bits 4004 en un sólo chip, su precio inicial es de $200.
1972	La Intel Corp. introduce su primer microprocesador de 8-bits 8008.
1973	Scelbi Computer Consulting ofrece su computadora Scelbi- 8H basada en el 8008 con 1 k. byte de memoria en $565.
1975	OTOÑO Sphere Corp. lanza la computadora Sphere I (6800, 4k bytes de RAM, monitor ROM, teclado, interfase de video, por $650)

AÑO	ACONTECIMIENTO
SEPT.	IBM anuncia la IBM , la primer computadora de bolsillo (con BASIC, 16 k bytes y sistema de almacenamiento en cartucho por $9 000).
1976	PRIMAVERA Texas Instruments anuncia su TMS 9000, el primer microprocesador de 16- bits
ABRIL	Se conforma la Apple Computer Inc.
DICIEMBRE	Shugart lanza su lectora de discos de 5 pulg. por $390.
1977	INVIERNO Ohio Scientific Instruments ofrece la primer microcomputadora con Microsoft (punto flotante) BASIC en ROM.
ABRIL	Jim Warren organiza el Primer Congreso de Computación de la Costa Oeste en San Francisco, donde se presentan la Apple II y la Commodore PET.
JUNIO	La Apple Computer Inc. lanza su primer anuncio (6502. 16k bytes de RAM, ROM y monitor integrado con 16 k bytes en ROM, teclado, interfase para caset, tarjeta principal con 8 ranuras de expansión, controles para juegos, interfase para gráficas-textos expandible a 48 k bytes en RAM) con un costo $ 2638
1978	MAYO Ken Bowles describe en la revista Byte el sistema operativo basado en lenguaje PASCAL UCSD.
DICIEMBRE	Epson America Inc. lanza la impresora de matriz de puntos MX-80, su alto perfeccionamiento y bajo precio apabulla a los competidores y los forza a bajar los precios en el mercado de las impresoras.
1979	Wayne Ratliff desarrolla el programa de base de datos Vulcan que mas tarde se convertirá en dBASE II. Video juegos como Pac Man y Space Invaders se convierten en la gran locura. Intel introduce el 8088 que se convertirá en el corazón de la IBM PC.
1980	FEBRERO Sinclair Research lanza su computadora ZX80, la primer microcomputadora con un costo menor a $200 (consta de 1k byte de RAM, 4k en ROM con BASIC integrado, teclado de membrana plástica). Su sucesor el ZX81, lo vendió mas tarde Timex en menos de $100 antes de que Timex dejara el mercado de las microcomputadoras. Satellite Software International, mas tarde WordPerfect Corp., anuncia la primer versión del WordPerfect para computadoras de datos

AÑO	ACONTECIMIENTO
	generales.
1981	AGOSTO IBM introduce la computadora personal IBM PC (8088, 64k bytes de RAM, 40 k bytes de ROM, lectora de discos de 5 pulg. con un costo de $ 3005) lo que legitima la industria de las microcomputadoras al resto del mundo y establece la preeminencia de la familia de procesadores Intel 8088 y el sistema operativo MS-DOS de Microsoft.
1982	VERANOEl lenguaje de programación LOGO se hace disponible para varias computadoras, mas notablemente para Apple II y T1-99/4A.
OCTUBRE	Lotus Development Corp. lanza su paquete de graficación y hoja de cálculo 1-2-3 con capacidad de manejo de listas para IBM PC. Su velocidad y capacidad permiten reemplazar al Visicalc y su combinación de varias funciones en un programa da comienzo al movimiento conocido como "software integrado" en microcomputadoras.
DICIEMBRE	Volition Systems presenta su primer implantación del lenguaje Modula-2
1983	FEBRERO IBM lanza su IBM PC XT. Aumenta a 10 mega byte en disco duro, 3 ranuras de expansión extra, una interfase de serie para el BASIC creado para la IBM PC. Con 128 k bytes de RAM y una lectora de discos y un costo de $ 4 995
ABRIL	Microsoft Corp, anuncia el Multi-Tool Word que mas tarde sólo se conocerá como Word.
MAYO	AT&T Information Systems presenta su sistema operativo UNIX System V.
OCTUBRE	Borland International Inc publica su Turbo Pascal para computadoras con procesadores CP/M y 8088. Su calidad, velocidad y bajo precio lo hacen inmediatamente estándar especialmente en el mundo de las IBM PC.
NOVIEMBRE	Microsoft Corp. presenta Windows, el software de ventanas múltiples para la IBM PC, pero que se hace disponible hasta el verano de 1985.
1984	ENERO Apple Computer Inc. introduce la Macintosh a $ 2495.
FEBRERO	Lotus anuncia su paquete Simphony a un precio de $695

AÑO	ACONTECIMIENTO
MARZO	Ashton-Tate lanza su paquete Framework por $695 en competencia con el Simphony
MAYO	La Apple Computer Inc. lanza la Apple IIc.
AGOSTO	La IBM presenta la IBM PC AT (80286, 256 k bytes en RAM, lectora de discos de 1.2 megas a un costo de $ 5 469) también lanza su red local PC.
	Hewlett Packard introduce la LaserJet y la HP110 una laptop 80C86
	Satellite Software International publica el WordPerfect para la IBMPC, Victor 9000, DEC Rainbow, Zenith Z-100 y Tandy 2000.
1985	Toshiba introduce su laptop T1100
	Aldus introduce el original Page Maker para la MAC 512-k byte.
	Ansa introduce Paradox que mas tarde comprará Borland.
	IBM introduce su red Token Ring
1986	Sperry y Burroughs se unifican en UNISYS
	Peter Norton anuncia su Norton Commander
	Microsoft introduce Works para la Mac.
1987	Apple introduce la Mac II y la Mac SE.
	IBM introduce PS/2 y OS/2, la primer IBM 386 (modelo 80).
	Commodore anuncia la Amiga 2000 y 5000
	Borland lanza el Quattro.
	IBM lanza la PS/2 modelo 25.
	Lotus firma un convenio con IBM para desarrollarle software por 10 años, comenzando con un 1-2-3/M.
1988	Intel anuncia su procesador 386SX.
	Ashton-Tate lanza el D-BASE IV.
	IBM y Microsoft lanzan su OS/2 1.1 con presentación de director.
1989	Novell anuncia su NetWare 386.
	Microsoft e IBM presentan su OS/2 1.2
	Borland anuncia su Turbo Pascal 5.5 con extensiones orientadas a objetos.
	Hewlett-Packard lanza su New Wave
	Después de muchos años Lotus lanza su hoja de calculo y procesador 3-D

AÑO	ACONTECIMIENTO
	Apple anuncia su Mac IIci y su laptop Mac Portable
1990	Microsoft anuncia su Windows 3.0
	Dragon Systems lanza su Dragon-Dictate con un sistema de reconocimiento de voz para 30 000 palabras, corre en 386 y tiene un costo de $ 9000
	Borland anuncia su Turbo C+ +.
	IBM anuncia la PS/1 que pretende romper el mercado de las PC para casa y oficina.

1.2 DEFINICION DE MODELO

Un modelo es una representación de un problema o de una situación real. Esta representación podemos hacerla de diferentes maneras y utilizando distintos recursos.

Independientemente de cómo y con qué hagamos nuestro modelo, en cualquier caso involucra un proceso de abstracción, que consiste basicamente en:

a) Seleccionar de la realidad, los elementos más importantes que intervienen en el problema y desechar aquellos que consideramos no juegan un papel determinante en el mismo.

b) Establecer con precisión las distintas relaciones que guardan entre si dichos elementos

Una vez realizado este proceso de abstracción estamos en condiciones de elaborar un modelo, dependiendo de cómo y con qué lo hagamos tomará distintas características. Construido el modelo, podemos manipular los elementos y sobre todo buscar posibles soluciones. Resolver el problema en el modelo significa haber contestado las siguientes preguntas:

a) ¿Existe solución? si la respuesta es negativa habremos terminado, el modelo construido no tiene solución, podemos replantearnos la pregunta y/o replantear el modelo. Si la respuesta es afirmativa la siguiente pregunta es:

b) ¿La solución es única? Si la respuesta es afirmativa habremos acabado, si resulta negativa, significa que existe más de una solución, la tercera pregunta es:

c) ¿Cuál de todas es la que más nos conviene? para contestar esta última muchas veces tenemos que volver a reflexionar sobre la realidad y/o sobre nuestro modelo, para establecer los criterios que nos permitan decir cual es mejor.

Después de resolver el problema en el modelo, podemos trasladar la solución encontrada a la realidad, este proceso recibe el nombre de aplicación.

Aunque, como hemos dicho anteriormente, al investigador de operaciones no le corresponde ni elegir la mejor solución (proponemos todas las soluciones posibles y señalamos las mejores), ni llevar a cabo ese proceso de aplicación. Lo que deseamos es que si se optó por una de nuestras soluciones propuestas al aplicarse a la realidad los resultados obtenidos correspondan a lo esperado.

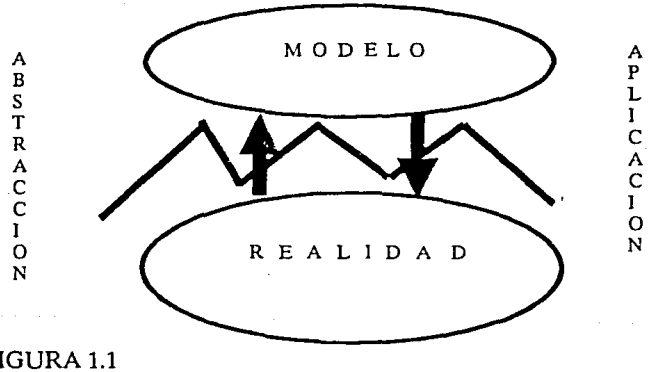


FIGURA 1.1

La Figura 1.1 ilustra esta relación entre los modelos y la realidad donde se aprecia con claridad un rompimiento entre la realidad y el modelo, este rompimiento se da en el proceso de abstracción al seleccionar los "elementos más importantes", y en la determinación de las reglas o leyes de relación entre dichos elementos, esto puede ocurrir por:

a) Una mala selección de los "principales elementos" y/o una mala determinación de las reglas de relación. Dicho de otra forma, el modelo

no corresponde a la realidad modelada.

b) Se elaboró un modelo adecuado, esto es, se hizo una selección adecuada de elementos, se determinaron con precisión las relaciones de interdependencia, pero su manipulación no fue del todo adecuada y las soluciones encontradas no resuelven el modelo, mucho menos la realidad.

c) Elaboración y manipulación correcta del modelo, sin embargo, la información para determinar los distintos parámetros, los elementos principales y/o las reglas de relacionales no corresponden a la realidad.

d) El modelo es adecuado y la manipulación fue correcta. La solución encontrada es satisfactoria, sin embargo, durante el tiempo de modelación y búsqueda de solución, la realidad ha cambiado. Elementos que eran importantes han dejado de serlo, relaciones de interdependencia entre elementos del problema se han modificado. La realidad es otra a la modelada por lo tanto el modelo no corresponde a la nueva realidad, a la nueva problemática.

En los primeros dos casos, se debe a errores de tipo humano, falta de destreza en la elaboración de modelos o de su manipulación, esto es, desconocimiento de la teoría necesaria para manejar el modelo. El tercer caso, se debe principalmente a que las bases de datos, de donde se obtiene la información, son alimentadas por sistemas muy rudimentarios que no verifican la información que se almacena, teniendo como resultado grandes bases de datos, parecidos a basureros de información donde hay que pepenar la información requerida. La labor de rescatar información de este tipo de bases de datos es muy laboriosa y no siempre es posible garantizar una alta confiabilidad de los datos. El último caso suele ser el menos frecuente, y es motivado por la dinámica propia de la realidad. Podría alegarse aquí, que un buen modelo debe considerar también esta propiedad cambiante de la realidad, al menos por hoy, esto no es posible, tal vez en el futuro se vuelva realidad la ficción de la psicohistoria novelada magistralmente por Issac Asimov en su trilogía trantoriana.

En cualquiera de estos casos, el decisor debe de tenerlos en cuenta y el investigador de operaciones debe de enfatizarlo. Es en el proceso de aplicación donde debe de manifestarse la experiencia y el sentido común de nuestros decisores.

1.3 MODELOS ANALITICOS Y HEURISTICOS.

MODELOS ANALITICOS.

Los modelos analíticos son aquellos que para su elaboración se utiliza el lenguaje proporcionado por las matemáticas y cuyas soluciones las buscamos apoyándonos en las teorías matemáticas.

En particular en la I. de O. el principal y sin lugar a dudas el modelo analítico más usado es el modelo de programación lineal.

Los conceptos matemáticos en que se apoya -las funciones lineales- son el resultado de un largo proceso de abstracción, aunque el origen de esta teoría este en fenómenos y problemáticas derivadas de la realidad, por su grado de abstracción, se ha despojado totalmente de las diversas características que le dieron origen.

En física por ejemplo, la expresión $d = v*t$ es un modelo analítico, que describe el comportamiento de un cuerpo que se mueve a velocidad constante durante un cierto intervalo de tiempo. Para llegar a esta expresión los físicos hicieron una serie de suposiciones sobre el modelo, dijeron: " esto es cierto bajo condiciones **ideales**, esto es sin considerar fricción, resistencia del aire - al vacío- , etc." Así este modelo describe una "realidad", que sólo podemos encontrar en un buen laboratorio.

Por lo tanto para llegar al modelo $d = v*t$, los físicos modelaron una realidad, despojada de todas sus características particulares como forma, color, materia del cuerpo, velocidad a que se mueve, etc. y esta porción de la realidad la han aislado totalmente de la realidad que la rodea, al considerar "condiciones ideales".

Quiere decir esto que el modelo $d = v*t$ ¿no describe la realidad?, la respuesta es no, a una realidad cotidiana, es decir que no nos enfrentemos con ella . Que describe una parte de la realidad si. Una porción que no se presenta sola en la realidad, sino acompañada de más elementos. **Esta parcialización de las descripciones de los modelos analíticos es su principal desventaja.**

Así en el caso del modelo de programación lineal, éste describe una situación ideal, determinista, continua y lineal, sin embargo la realidad dista mucho de ser una situación ideal, pero comprender y sobre todo predecir el comportamiento de una realidad, aunque sea bajo la suposición de condiciones ideales nos ayuda mucho.

La ventaja de los modelos analíticos, es su simplicidad y exactitud, los modelos

analíticos -si están bien planteados desde luego- no dejan lugar a dudas en sus resultados.

No todos los modelos analíticos son deterministas, la probabilidad es la parte de las matemáticas que tuvo su origen en la descripción estocástica del mundo. Las variables aleatorias, como toda función, describen o pretenden describir el comportamiento de un fenómeno de la realidad. Asociar el comportamiento de un ente con una variable aleatoria, que tenga una expresión analítica es un verdadero problema y no en todos los casos es posible. Sin embargo trabajar con una variable aleatoria que tenga una expresión analítica conocida, simplifica mucho nuestros cálculos. En la teoría de la probabilidad la mayor dificultad radica en saber interpretar los resultados a que se llegan.

La mayoría de los modelos analíticos son susceptibles de programarse en una computadora. En la actualidad existe un buen número de paquetes de cómputo que resuelven el modelo de programación lineal, a través de derivaciones del método simplex, lo que ha popularizado su utilización. Sin embargo actualmente se desarrollan nuevos métodos para la solución del modelo de programación lineal, como el algoritmo de Khachian que tiene la propiedad de que el número de operaciones nunca excede una cota polinomial fija, al menos en el peor de los casos. Este algoritmo mantiene una solución desconocida -el vector que minimiza o maximiza cx- dentro de una serie de elipsoides que se contraen en un punto. Ese punto es la solución. Otro algoritmo más reciente llamado el Metodo de Karmarkar también opera dentro de un conjunto factible - mientras que el método simplex opera en los extremos. La solución se encuentra en tiempo polinomial y se dice que es cincuenta veces mas rápido que el simplex.

La programación de los modelos analíticos ha dado una nueva connotación al concepto de algoritmo, cuyo origen lo encontramos en Euclides (año 300 antes de nuestra era). El concepto de algoritmo, podemos resumirlo como un proceso formado por:

a) Etapa de preparación.

b) Ciclo iterativo.

c) Criterio de terminación.

Un algoritmo asegura una solución óptima, en un número finito de pasos y debe ir acompañado de su correspondiente demostración.

Podemos concluir que en general, los modelos analíticos tienen las ventajas de la exactitud y la desventaja de la parcialización del modelo.

MODELOS HEURISTICOS.

La heurística o heurética, o "ars inveniendi", tal es el nombre de esta ciencia que tiene por objeto el estudio de las reglas y de los métodos del descubrimiento y la invención. Algunos comentarios a cerca de la heurística los encontramos en **Papus** (siglo III de nuestra era), comentando a Euclides. Los ensayos más conocidos sobre la construcción de un sistema heurístico son debidos a **Descartes (1596-1650)** y **Leibniz (1646-1716)**, ambos filósofos y matemáticos brillantes. Igualmente debemos a **Bernard Bolzano (1781-1848)** una exposición sobre heurística detallada y notable. Sin embargo, quien retoma la heurística y le da la connotación actual fue **George Polya** (1887-), quien en 1945 publicó un pequeño libro **How to solve it** (Cómo resolverlo), cuya traducción al español fue Cómo plantear y resolver problemas. La portada del libro promete "Un sistema de pensar que puede ayudarle a resolver cualquier problema". La pretensión resulta excesiva, pero el libro es sumamente instructivo. Polya define su libro como heurístico, palabra que significa "que sirve para descubrir". Este libro considera cuestiones generales con los que se enfrentan los que resuelven problemas y propone métodos generales para tratarlos. Propone una serie de preguntas para cada etapa de su metodología de solución.

Dentro de la I. de O. se consideró inicialmente a la modelación heurística como los procedimientos que se siguen paso a paso y que aseguran, mediante un número finito de ellos, alcanzar una solución satisfactoria. Poco difiere esta acepción de la de algoritmo, ya que difieren exclusivamente en que ésta, no nos garantiza una solución óptima y desde luego no requiere de una demostración, además hay algoritmos que pueden garantizar un porcentaje alto de acercamiento al óptimo. Así la modelación heurística nos deja manos libres en la búsqueda de soluciones a diversas problemáticas y sobre todo nos permita modelar situaciones que analíticamente no pueden modelarse, o su modelación puede ser muy costosa.

Resumiendo la heurística nos da oportunidad de modelar problemáticas más complejas.

1.4 MODELACION DE SISTEMAS.

Hasta aquí hemos hablado de la modelación de problemáticas, hemos hecho a un lado el concepto de sistema, sin embargo, la importancia misma del concepto nos obliga a darle un trato especial.

"Sistema", es uno de los conceptos relevantes producto de esta revolución tecnológica. Su impacto y difusión ha inundado todas las áreas del conocimiento

humano, se ha presentado como técnica, como metodología y finalmente como una filosofía. Más que dar una amplia discusión al respecto, el objetivo de estas lineas es delimitar algunas de sus características que para nuestros objetivos (la modelación de sistemas) resultan valiosos.

La definición más popular de sistema es "Un conjunto de elementos interrelacionados que constituyen una totalidad", esta definición es tan vaga que en ella tienen cabida desde un átomo, hasta el universo mismo, desde una célula, hasta el organismo vivo más complejo, etc. Así cualquier problemática de las que nos hemos referido arriba al hablar de modelación, nos estábamos refiriendo a sistemas. Sin embargo viene a constituir un marco teórico que nos ayuda a entender el rompimiento que se da entre la realidad y el modelo, a continuación daremos un resumen de sus principales características:

1. COMPLEJIDAD. El enfoque de sistemas parte del reconocimiento de que la realidad es compleja. Esta complejidad se debe a que los elementos del objeto a modelar están íntimamente interelacionados y a que el objeto mismo interactúa en el medio ambiente con otros objetos. (fig.1.2)

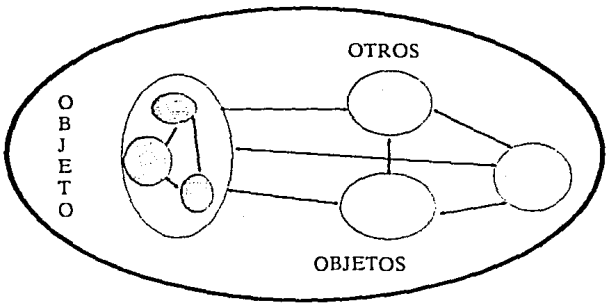


FIGURA 1.2

El reconocer esta complejidad, lleva consigo el aceptar que el funcionamiento adecuado del objeto, esta más allá del correcto desempeño de los elementos que lo componen, ya que también influye la manera en que estos interactúan entre sí y con su entorno.

Este argumento es válido para cualquier sistema, y su peso depende del problema mismo a modelar. La modelación de un sistema puede requerir el concurso de especialistas de diversas disciplinas, lo que demanda una visión interdisciplinaria.

Cualquiera que sea la realidad a modelar, por pequeña y simple que nos parezca,

tiene cierto grado de complejidad. No esta por demás reconocer que este trabajo esta dirigido a resolver problemáticas modestas

2. TOTALIDAD. El enfoque de sistemas reconoce la necesidad de tener una visión global o total de la problemática a modelar. Sin lugar a dudas este es siempre el primer paso y significa, además de definir el sistema y su problemática, establecer la finalidad del modelo. El no tener esta visión global del problema, acarrea pérdida de tiempo y recursos, es como "tirar palos de ciego".

Adquirir esta visión global del problema no es fácil, depende de la naturaleza misma de la problemática a modelar, algunas veces esto es tan difícil que nos parece imposible. Por ejemplo los problemas sociales.

3. ANALISIS Y SINTESIS. El enfoque de sistemas, no ignora ni se contrapone a lo que algunos autores han denominado enfoque analítico el cual propone:

a) Aislar y dividir en partes el todo

b) Entender el funcionamiento de cada una de las partes.

c) Reunir el conocimiento de las partes para entender el comportamiento y propiedades del todo.

Para entender cada una de las partes, puede procederse de la misma forma tantas veces como sea necesario.

El enfoque sistémico, sostiene que cuando en el sistema a modelar tiene cierto grado de complejidad, como relaciones no lineales o expresiones no analíticas, este procedimiento se complica e invalida. Ante esta situación el enfoque sistémico añade el estudio de las partes que interaccionan como un todo. Así el enfoque sistémico propone las siguientes características al proceso de síntesis, entendido éste como el proceso de integrar el conocimiento de las partes para entender el todo.

a) Las propiedades o el comportamiento de cada elemento del conjunto tiene un efecto en las propiedades o comportamiento del todo.

b) Las propiedades o el comportamiento de cada elemento y la forma en que afecta al todo depende de las propiedades y del comportamiento de al menos otro elemento del conjunto.

c) Cada subgrupo posible exhibe las dos propiedades anteriores.

Así, el enfoque sistémico llama a la creación de nuevas técnicas y métodos, cuya característica básica radica en que el conjunto de elementos y relaciones deben estudiarse bajo esta perspectiva de totalidad.

EL MODELO DIAMANTE.

Durante la década de los 70's Ian I. Mitroff, junto con otros autores, desarrollaron un modelo cualitativo de solución de problemas, llamado modelo diamante, en el cual en forma un tanto esquemática divide el proceso de solución de problemas en cuatro fases, que tienen un orden de desarrollo y que requieren de retroalimentarse entre sí, me permito resumir dicho modelo porque nos sirva como marco de referencia en la definición de modelo de simulación.

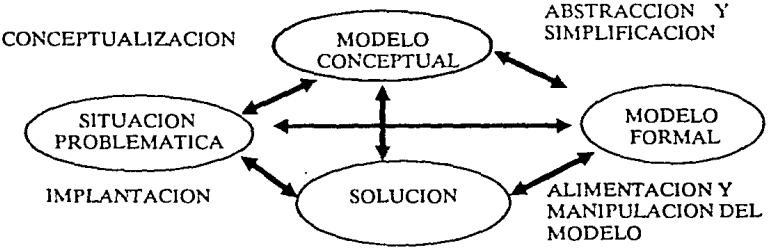


FIGURA 1.3

A. Situación Problemática: Consiste en el reconocimiento del problema, es el reconocimiento de que algo anda mal o el sentimiento de que algo puede mejorarse.

B. Modelo Conceptual: Consiste en lo que hemos llamado definición del problema, esto es, establecer la estructura del problema, delimitar el área de interés y establecer los elementos relevantes y las relaciones que guardan entre si.

C. Modelo Formal: Generalmente consiste en un modelo matemático, un conjunto de ecuaciones, etc.

D. Solución: Esta actividad es la integración de los distintos modos de acción, es la integración de las estrategias de implantar en la realidad el sistema.

Los modelos de simulación, se aproximan más a lo que, en el modelo diamante, se define como modelo conceptual y al que hemos denominado modelo heurístico, que permite integrar en él uno o varios modelos analíticos (o formales).

CAPITULO II

SIMULACION DIGITAL.

No es digno de hombres excelentes perder como esclavos, horas y horas haciendo cálculos que, con facilidad, un cualquiera podría hacer por medio de máquinas.

Barón Gottfried Wilhelm Von Leibnitz.

La simulación digital, es la modelación de cierto tipo de sistemas utilizando un computador digital. Inicialmente empieza a utilizarse en la solución de problemas de lineas de espera, cuya solución analítica no es posible, por lo que podemos afirmar que vino a apoyar a la modelación analítica.

Una de las características de estos primeros modelos simulados es que son aleatorios y dinámicos. Tal es el caso de los modelos de lineas de espera que se desarrollan a lo largo de un intervalo de tiempo dado. Por ejemplo, el modelo clásico consiste en una serie de clientes que llegan a intervalos de tiempo con una determinada distribución a un modulo de servicio en donde si no se está atendiendo a nadie pasa a ser atendido, en caso contrario espera formando una línea de espera, el tiempo de servicio también es una variable aleatoria. La pretensión usual del modelo es estimar los tiempos medios de permanencia de los clientes, el tamaño máximo de la línea de espera y el porcentaje de tiempo de ocupación del modulo de servicio, para que con ésta información se puedan hacer los ajustes convenientes y mejorar por ejemplo el tiempo de servicio.

El primer problema que tuvieron que enfrentar estos modelos fue la generación de números aleatorios con una función de distribución determinada, este problema se reduce al de generar una serie de números aleatorios con distribución uniforme en el intervalo cero - uno y calcular la inversa de la función acumulada de probabilidad de la distribución de números deseada, esta técnica se ilustra en el anexo 1 y tiene la ventaja de poder generar series de números con distribuciones empíricas.

La técnica de simulación consiste básicamente en reproducir varios ciclos de vida del sistema y sacar las estadísticas respectivas, por ejemplo en el modelo clásico que se dio en el párrafo anterior, un ciclo de vida del sistema sería un día de labores, de donde resolverlo por simulación seria simular varios días de servicio y sacar estadísticas de los comportamientos observados.

Este capítulo se desarrolla como sigue: En la primera sección se da una breve introducción histórica del desarrollo de la simulación y se dan algunas definiciones. En la segunda se presentan algunas definiciones y conceptos básicos de simulación. En la tercera se describe en términos generales como se desarrolla una simulación, dentro de la cual se resaltan dos puntos: la validación del modelo simulado así como su análisis de sensibilidad e implantación. En la cuarta se presentan algunas ventajas y desventajas de hacer simulación. En la quinta sección se describen los lenguajes de propósito especial así como una clasificación de ellos y finalmente en la sexta se plantean algunas lineas de desarrollo.

2.1 INTRODUCCION.

Durante la década de los 60's, la técnica de simulación sirvió básicamente como apoyo a las técnicas analíticas en la solución de modelos de subsistemas. Su aplicación resultaba costosa en términos del tiempo de máquina requerido por cada corrida. A pesar de esto fue en esta época cuando se desarrolló la primera versión del GPSS, lenguaje de programación de propósito especifico, para simulación de sistemas discretos.

La simulación digital, desde sus inicios dejó ver un enorme potencial en la solución de problemas, su principal limitación durante sus primeros 20 años de desarrollo, como ya se ha dicho fueron los altos costos de los equipos de cómputo. Con la aparición de las microcomputadoras, su posterior desarrollo y abaratamiento, se sentaron las bases materiales para un gradual y pujante desarrollo de la simulacion digital.

En la década de los 70's, la simulación digital, empezó a ser usada a mayor escala

y da aportaciones en:

- El desarrollo de la teoría de la I. de O.
- En la obtención de información en cuanto al funcionamiento de un sistema y
- En la validación de modelos analíticos.

Un cuarto uso de la simulación es en optimización. Ya en 1980 Hanssmann et al, da una discusión detallada de la utilización de modelos de simulación en optimización. Éste autor distingue cuatro métodos de optimización. Cada método optimiza primero un modelo analítico aproximado y después utiliza los valores encontrados como parámetros en el modelo de simulación.

Los cuatro métodos difieren en la forma en que se usan los parámetros derivados del modelo analítico. Estos son:

a) Para determinar resultados de los parámetros trabajados.

b) Como punto de inicio de un proceso de búsqueda en un procedimiento de solución por simulación.

c) Para determinar resultados de los parámetros trabajados que retroalimentarán el procedimiento de solución del modelo de simulación para su uso en reoptimización del modelo analítico. El nuevo conjunto de valores solución se usa entonces como nuevos parámetros en el modelo de simulación y así se continúa iterando entre ambos modelos hasta obtener resultados esperados.

d) Igual que en c), con la diferencia de que en cada iteración en el modelo de simulación se aplica un procedimiento de búsqueda como en b).

Los métodos c) y d) permiten que la iteración de soluciones comience con cualquiera de los dos modelos. Si la sucesión comienza con el modelo de simulación, el analista debe seleccionar los parámetros iniciales en el simulador.

Fetter and Thompson reportaron el uso de este tipo de modelación (4.c), en un estudio para la asignación y operación de un hospital, iterando entre el uso de un modelo de programación lineal y un modelo de simulación. Con el modelo de programación ubicaban las camas en las diferentes unidades del hospital y con el simulador determinaban la política del uso de las camas, esto es, las reglas operacionales de funcionamiento del hospital.

Con todo lo referido anteriormente podemos intentar dar una definición de simulación como un método usado para estudiar la dinámica de sistemas. Donde por sistema entendemos un grupo de unidades que operan en forma interrelacionada. A menudo el propósito del estudio de los sistemas es entender de manera más exacta la operación total de las unidades que lo conforman. La simulación proporciona una descripción del funcionamiento del sistema en un periodo de tiempo.

Otra definición es la dada por Shannon:

Simulación es el proceso de diseñar un modelo de un sistema real y realizar experimentos con él para entender el comportamiento del sistema o evaluar varias estrategias (dentro de los límites impuestos por un criterio o por un conjunto de criterios) para la operación del sistema.

En consecuencia entendemos que el proceso de simulación incluye tanto la construcción del modelo como su uso analítico para estudiar un problema. Ya en el capítulo 1 esbozamos algunas ideas y definición en torno a la modelación.

Es entonces de ésta forma que no restringimos nuestra definición de simulación a los experimentos realizados sobre modelos de computadoras. Muchas simulaciones útiles pueden realizarse y se realizan con sólo lápiz y papel o con la ayuda de una calculadora. El modelado de la simulación es, por tanto, una metodología aplicada y experimental que intenta:

a) Describir el comportamiento de sistemas.

b) Postular teorías o hipótesis que expliquen el comportamiento observado.

c) Usar estas teorías para predecir un comportamiento futuro, es decir, los efectos que se producirán mediante cambios en el sistema o en su método de operación.

A diferencia de la mayoría de las tecnologías, las cuales pueden clasificarse de acuerdo con la disciplina (por ejemplo, física o química) de la que se originan, la simulación es aplicable a todas las disciplinas. La simulación como la conocemos recibió su impulso original de los programas aeroespaciales, pero incluso un examen superficial de la literatura sobre este tema indica el amplio campo de sus aplicaciones actuales. Por ejemplo, en administración, economía, educación, política, transporte e inumerables áreas.

2.2 DEFINICIONES Y CONCEPTOS BASICOS DE SIMULACION.

A continuación se presentan una serie de definiciones que, a pesar de las limitaciones que se puedan encontrar en ellas, nos permitirán trabajar con precisión las técnicas de simulación. No se pretende dar una discusión de los conceptos aquí definidos, se trata de dejar claros algunos rasgo de dichos conceptos.

Definición 2.1 **Simulación** es una técnica experimental para resolver problemas que nos permite tener un modo efectivo de probar, manejar y evaluar un sistema propuesto sin tener acción directa sobre el sistema real.

Definición 2.2 **Sistema** es un conjunto de elementos unidos entre si por medio de una interacción o interdependencia regular Como se ilustra en la figura 2.1

Definición 2.3 **Entidad** es un elemento que nos interesa en el sistema.

Definición 2.4 **Atributo** es una descripción o definición de las propiedades de las entidades.

Definición 2.5 **Actividad** proceso que causa cambios en el sistema.

La tabla 2.1 ilustra las definiciones anteriores.

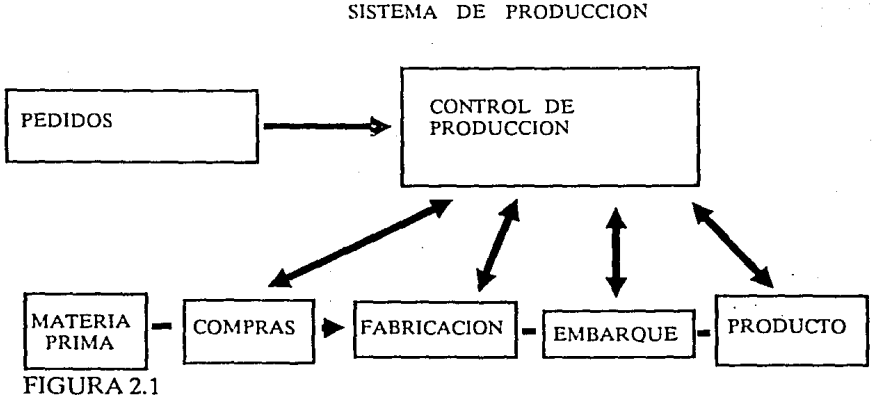


FIGURA 2.1

Definición 2.6 **Estado del sistema** es la descripción de entidades, atributos y actividades del sistema en un momento dado (es una fotografía).

Sistema	Entidad	Atributo	Actividades
Banco	Cliente, ventanilla, terminal de computadora	Edo. de cta, tamaño de la cola, dispo_ nibilidad.	Depósito, consulta, retiro, etc.
Computadora	Discos, impresora, memoria, puertos	Capacidad de memoria, velocidad de acceso	Entrada de datos, búsqueda de archivos, etc
Metro.	pasajeros, estaciones, trenes	longitud entre estaciones, capacidad,	Llegadas a estación de trenes y pa_ sajeros, etc.
Super.	clientes, cuentas, cajas.	cuantos com_ pran, cuantos hay.	pagar, llegada de clientes.

Tabla 2.1

Definición 2.7 **Modelo** es una representación del sistema o parte del sistema que queremos estudiar (entidades, atributos, actividades).

Definición 2.8 **Actividad Endógena** es aquella que se genera dentro del mismo sistema.

Definición 2.9 **Actividad Exógena** es aquella que proviene del medio exterior.

Definición 2.10 **Un sistema es cerrado** si no tiene actividades exógenas y es **abierto** si al menos tiene una.

Ejemplo 2.1

Los tiempos de llegada a un estacionamiento son endógenos si con base en sus atributos los generamos dentro del sistema. Son exógenos si es la información histórica, la que ocupamos para la asignación de atributos.

Definición 2.11 **Un sistema continuo** es aquel en el que los cambios son suaves (smooth) **es discreto**, si los cambios son puntuales en el tiempo. Se trabajará con sistemas discretos, discretizando si es necesario.

Definición 2.12 **Modelo** es una representación de un objeto, un sistema, o idea, de forma diferente a la de la identidad misma.

Los modelos dependiendo de la finalidad que se persiguen los podemos clasificar en:

Descriptivo.- Representa el sistema sin tener acción sobre el.

Predictivo.- Representa el comportamiento del sistema.

Explicativo.- Sirve para tratar de entender el comportamiento.

Otra clasificación, ahora atendiendo la constitución del modelo es:

Icónicos.- Las característica o propiedades se representan por si mismas (también llamados físicos).

Analógicos.- Las propiedades son representadas por otras equivalentes (modelos hidráulicos para representar flujo de transito).

Simbólicos.- Las propiedades se representan por símbolos y las relaciones entre ellas por relaciones entre los símbolos (modelos matemáticos).

Lógico.- Esta dado por elementos lógicos que al irse siguiendo representan el comportamiento del sistema (diagramas de flujo).

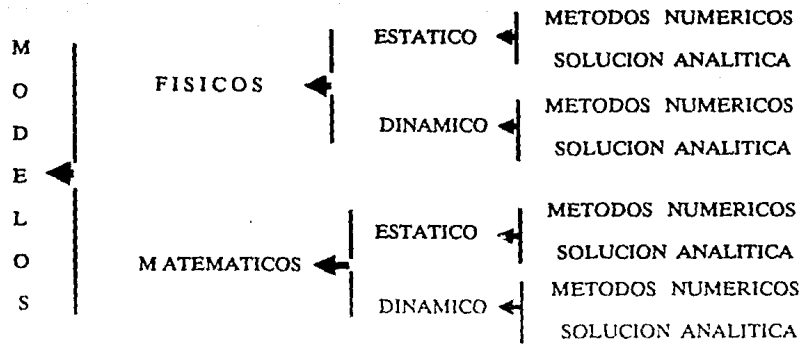
La figura 2.2 es una subclasificación de los modelos físicos (icónicos) y de los matemáticos o simbólicos, atendiendo a las características de sus variables.

Definición 2.13 Un modelo es **estático** si no esta en función del tiempo.

Nos interesan las relaciones entre los atributos cuando el sistema esta en equilibrio. Por ejemplo fórmulas para determinar cargas, modelos econométricos que no dependen del tiempo, etc..

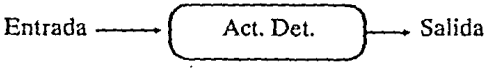
Definición 2.14 Un sistema **dinámico** representa cambios en el tiempo.

por ejemplo, considere f(x,y,z,t) una ecuación de movimiento, si f es sencilla



podemos encontrar una solución analítica, de lo contrario obtenemos una solución numérica que da los valores de x, si en ciertos tiempos, en cualquier caso, se trata de un modelo dinámico.

Definición 2.15 **Actividad determinística**: aquella cuyo resultado está totalmente determinado por los datos de entrada.



Definición 2.16 **Modelo determinístico**: formado exclusivamente por actividades determinísticas. se puede seguir el modelo siguiendo las actividades que lo forman.

Definición 2.17 **Variable estocástica**: es una variable que corresponde a una actividad aleatoria. No se conoce la secuencia de los valores pero si el rango de ellos y su probabilidad.

2.3 PROCESO DE SIMULACION DE SISTEMAS.

Resultaría limitante dar una serie de pasos a seguir al resolver problemas de simulación, y decir que son los únicos, los distintos autores que escriben al respecto proponen de acuerdo a su experiencia personal, a continuación reproducimos los propuestos por el Maestro en I. de O. Leonard Rapoport en sus cátedras de la DEPFI:

- 1.-Definición del problema.
- 2.-Planeación del estudio.
- 3.-Revisión y obtención de datos.
- 4.-Formulación y evaluación del modelo matemático.
- 5.-Formulación del programa de computo.
- 6.-Validación.
- 7.-Diseño de experimentos.
- 8.-Ejecución de la simulación.

1.- **Definición del problema:-** Parece obvio, pero lo primero que se debe hacer es definir que es lo que se quiere estudiar, ¿Que problemática pretende resolverse?, ¿Que se persigue?. En no pocas ocasiones se nos invita a participar en proyectos, cuya problemática ni siquiera esta bien definida, de aquí que el primer paso consista en definirla.

2.- **Planeación del estudio:-** Una vez superado el punto anterior, hay que determinar: el tiempo con el que contamos, los recursos de que se dispone, gente, computadora, dinero, etc. Una vez determinados los recursos hay que definir los alcances y planear el tiempo de realización del trabajo.

3.- **Revisión y obtención de datos:-** ¿ Con que información contamos?. Hasta hace pocos años, el principal problema consistía en que no existía información almacenada, había que diseñar estrategias para su obtención y sobre todo ser lo suficientemente creativos para buscar fuentes alternas de información. Para ejemplificar lo anterior basta recordar cómo se construyó la matriz origen-detino para el estudio de

la línea 2 del metro, (tengo entendido que para la línea 1, no se realizó un estudio tan detallado como para las subsecuentes líneas). Hay que recordar que a principios de la decada de los 70, no se realizaban estudios sistemáticos sobre el movimiento de los capitalinos. Planear un estudio que nos proporcionara la información necesaria para su elaboración, involucraba una serie de encuestas, cuyo diseño, aplicación , depuración y procesamiento resultaba sumamente costo. El problema se resolvió, acudiendo a los archivos del IMSS: los trabajadores se registran en la clínica más cercana a su hogar y entre los datos que proporciona es el de su domicilio y la empresa en que labora. El que propuso esa idea, provocó un ahorro de no pocos millones de pesos, éste es un bonito ejemplo de una idea productiva. En la actualidad y al menos durante algunos años más, el principal problema consiste en depurar la información almacenada en las computadoras de la mayoría de nuestras empresas, ya que, la mayoría de los sistemas no verifican la información que reciben, lo que ocasiona verdaderos basureros de información, de donde hay que rescatar lo rescatable, un trabajo de pepenadores de información, que requiere una gran habilidad y sobre todo creatividad para realizar esta labor.

4.- **Formulación y evaluación del modelo.-** Estructurar el modelo: definir variables, interrelación de variables, distribuciones de probabilidad, etc. En esta etapa juega un papel importante, el que conoce el sistema a simular. Con frecuencia al ir desarrollando el punto 5, se vuelve a este punto.

5.- **Formulación del programa de cómputo:** Programar y probar el modelo. Los objetivos del modelo estan muy ligados al lenguaje de programación que se usará. Este punto descansa sobre todo en el programador.

6.- **Validación y calibración:-** ¿ El programa que representa el modelo, refleja adecuadamente el sistema?. Alimentar al modelo con ciertas condiciones conocidas del sistema, comparar los resultados obtenidos con los esperados y mover los parámetros de calibración del sistema, hasta obtener los resultados esperados, es un proceso de pruebas y ajustes, que nos conduce con frecuencia a un ciclo, donde volvemos al punto 4, es decir a formular nuevamente el modelo, reprogramar los cambios y validar el modelo. Así hasta que el modelo corresponda con la realidad.

7.- **Diseño de experimentos:-** diseñar las diferentes corridas que se quieran hacer con el modelo, para obtener ciertos resultados que al analizarlos den luz de lo que se quiere.

8.- **Ejecución de la simulación:** corrida en computadora.

Estos pasos los podemos agrupar y representar mediante el diagrama de flujo de la figura 2.3, que nos ayudará a entender como se relacionan:

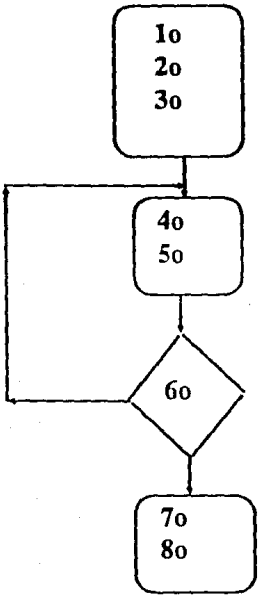


FIGURA 2.3

Con los pasos anteriores se supone que el problema puede resolverse de una mejor manera mediante la simulación. Como ya se indicó, ésta quizá no sea la forma mas efectiva. A menudo se ha dicho que la simulación es un planteamiento aproximado o un último recurso para resolver problemas. En realidad, es cierto que cuando un problema puede reducirse a un modelo simple y resolverse analíticamente, la simulación no es necesaria. Debería investigarse todas las herramientas disponibles para manejar cada problema y optimizar entre los resultados y el costo. También un debería convencerse de que un modelo analítico simple es inadecuado, antes de considerar la posibilidad de una simulación.

Debido a que es necesario y conveniente ajustar la herramienta al problema, las decisiones referentes a qué herramienta o método usar deben seguir la formulación del problema. La decisión de usar la simulación no debe considerarse como irrevocable. Conforme se obtienen más datos y se entiende más el problema, la validez del uso de la simulación debe revaluarse. Debido a que con frecuencia intervienen grandes computadoras y muestras grandes, el costo de una simulación

es casi siempre alto comparado con la resolución de pequeño problema analítico. El costo probable y el tiempo de la simulación siempre deberán compararse con el valor de la información que producirán.

2.3.1 VALIDACION DEL MODELO

Un modelo debería crearse exclusivamente para un propósito específico, y su veracidad o validez debería evaluarse solamente en términos de ese propósito. Nuestra meta es generar un modelo que cree los mismos problemas y características de comportamiento que las del proceso o sistema que se está estudiando. Evaluar un modelo significa desarrollar un nivel aceptable de confianza de modo que las inferencias obtenidas del comportamiento del modelo sean correctas y aplicables al sistema del mundo real.

Así la validación es el proceso de llevar a un nivel aceptable la confianza del usuario referente a que cualquier inferencia acerca de un sistema que se derive de la simulación sea correcta. Es imposible probar que cualquier simulador es un modelo correcto o verdadero del sistema real. Por fortuna, rara vez nos interesamos en probar la "verdad" de un modelo. En vez de ésto, principalmente nos importa la validación de los discernimientos que hemos obtenido u obtendremos a partir de la simulación. Por lo tanto, nos importa la utilidad operativa del modelo y no la verdad de su estructura.

La validación del modelo es sumamente importante debido a que los simuladores se ven reales y tanto los modeladores como los usuarios los encuentran fácilmente creíbles. Lamentablemente, con frecuencia los simuladores ocultan sus suposiciones del observador casual y algunas veces del modelador. Por lo tanto, si la validación no se lleva a cabo cuidadosa y detalladamente, pueden aceptarse resultados erróneos con consecuencias desastrosas.

No existe una prueba de validación, en lugar de eso, el experimentador debe realizar pruebas a lo largo del proceso de desarrollo del modelo a fin de crear confianza. Se pueden usar tres pruebas para validar un modelo:

Primera: debemos cerciorarnos de que el modelo tenga validez en forma general, por ejemplo: ¿Es posible que el modelo dé respuestas absurdas si se lleva a los parámetros a valores extremos?. O también podemos preguntar si los resultados del modelo parecen razonables, esto se puede hacer mediante la creación de un panel de expertos de dos tipos: en software y en el sistema real de lo que se esté simulando.

El segundo método de validación es la prueba de suposición y el tercero es la prueba de transformaciones de entrada-salida. Estas dos últimas conllevan el uso

de pruebas estadísticas de medias y varianzas, regresión, análisis de factores, análisis espectral, autocorrelación, ji-cuadrada, y pruebas no paramétricas. Debido a que cada una de estas pruebas estadísticas hacen suposiciones acerca del proceso fundamental, el uso de éstas da origen a las preguntas de validez. Algunas pruebas estadísticas requieren menos suposiciones que otras, pero en general, la potencia de la prueba disminuye conforme se atenúan las suposiciones.

Fishman y Kiviat dividen la evaluación de las simulaciones en tres categorías:

1. Verificación - asegurándose de que el modelo se comporta de la manera que el experimentador desee.

2. Validación - probando la concordancia entre el desempeño del modelo y el desempeño del sistema real y

3. Análisis del problema - dibujando las inferencias estadísticamente significativas a partir de los datos generados por la simulación de computadoras.

Sin embargo, la realización de esta prueba padece de los problemas estándar de la investigación empírica: pequeñas muestras debido al alto costo de los datos, datos que están muy agrupados y datos de validez dudosa.

Por lo tanto la pregunta acerca de la validación tiene dos facetas: la determinación de que el modelo se comporte igual que el sistema del mundo real; y la validación de que las inferencias que se obtienen de los experimentos con el modelo sean válidas o correctas. Conceptualmente, ambos puntos se transforman en el problema estándar de decisión referente al equilibrio que debe existir entre el costo de cada acción contra el valor de la información aumentada y las consecuencias de obtener conclusiones erróneas.

2.3.2 ANALISIS DE SENSIBILIDAD E IMPLANTACION

El análisis de sensibilidad es uno de los conceptos más importantes en los modelos de simulación, por esto queremos decir, determinar la sensibilidad de nuestras respuestas finales a los valores de los parámetros que se usaron. Usualmente el análisis de sensibilidad consiste en la variación sistemática de los valores de los parámetros sobre algún intervalo de interés y en la observación del efecto en la respuesta del modelo. En casi cualquier modelo de simulación, gran parte de las variables establecidas se basan en datos sumamente dudosos. En la mayoría de los casos, estos valores se pueden determinar sólo con base en las conjeturas del personal experimentado o en un análisis muy superficial de datos mínimos. En

consecuencia, es vital determinar el grado de sensibilidad de los resultados para los valores que se usaron. Si la respuesta cambia enormemente con pequeñas variaciones en los valores de algunos de estos parámetros, esto puede proporcionar la motivación y justificación para asignar más tiempo y dinero para obtener estimaciones más precisas. Además si los resultados no cambian sobre las amplias fluctuaciones en los valores del parámetro, no se necesita un esfuerzo adicional.

En la simulación, el modelador tiene un control absoluto sobre su modelo de aquí que el análisis de sensibilidad sea importante ya que se pueden manipular facilmente los parámetros del modelo.

Finalmente se necesitan otros dos elementos a incluirse en cualquier proyecto de simulación y son la implantación y la documentación. Es importante que se implante el modelo ya que una de las principales causas de fracaso en la investigación de operaciones y en proyectos de ciencias administrativas se debe al mal entendimiento de los resultados por parte del usuario y a la falta de implantación.

La documentación se relaciona estrechamente con la implantación. La documentación cuidadosa y completa del desarrollo y la operación del modelo pueden incrementar notablemente su vida útil y sus oportunidades para una exitosa implantación. La buena documentación facilita la modificación y asegura que el modelo pueda usarse a pesar de que los servicios de los que se encargan del desarrollo ya no estén disponibles.Así como poder usarse en caso de realizar proyectos en el futuro que la requieran.

2.4 VENTAJAS Y DESVENTAJAS DE LA SIMULACION.

Todos los modelos de simulación se llaman modelos de entrada-salida. Es decir, ellos producen la salida del sistema si se les da la entrada a sus subsistemas interactuantes. Por tanto, los modelos de simulación se "corren" en vez de "resolverse" a fin de obtener la información o los resultados deseados. Son incapaces de generar una solución por sí mismos en el sentido de los modelos analíticos; sólo pueden servir como herramienta para el análisis del comportamiento de un sistema en condiciones especificadas por el experimentador. Por tanto, la simulación no es una teoría, sino una metodología de resolución de problemas. Además la simulación es sólo uno de varios planteamientos valiosos para resolver problemas que están disponibles para el analista de sistemas. Debido a que es necesario y deseable ajustar la herramienta o técnica al problema en vez de lo contrario, esto hace que surja la pregunta ¿Cuándo es útil la simulación?

Hemos definido la simulación como la experimentación con un modelo de un sistema real. Un problema experimental aparece cuando surge la necesidad de cierta información específica acerca de un sistema, la cual no está disponible en las fuentes existentes conocidas . Sin embargo algunas veces las desventajas de la experimentación directa son muy grandes.

1. Puede interrumpir las operaciones de la compañía.

2. Si la gente es parte integral del sistema, pueden verse afectados los resultados, el hecho de que se observe a la gente puede modificar su comportamiento.

3. Mantener las mismas condiciones operativas para cada repetición o corrida del experimento, puede ser muy difícil.

4. El obtener el mismo tamaño de la muestra (y por tanto importancia estadística) puede requerir más tiempo y ser más costoso.

5. Quizás no pueda ser posible explorar muchos tipos de alternativas en la experimentación del mundo real.

Por lo tanto, el analista debe considerar el uso de la simulación cuando existan una o más de las siguientes condiciones:

1. No existe una completa formulación matemática del problema o los métodos analíticos para resolver el modelo matemático no se han desarrollado aún. Muchos modelos de lineas de espera corresponden a ésta categoría.

2. Los métodos analíticos están disponibles, pero los procedimientos matemáticos son tan complejos y difíciles, que la simulación proporciona un método más simple de solución.

3. Las soluciones analíticas existen y son posibles, pero están más allá de la habilidad matemática del personal disponible. El costo del diseño, la prueba y la corrida de una simulación debe entonces evaluarse contra el costo de obtener ayuda externa.

4. Se desea observar el trayecto histórico simulado del proceso sobre un periodo, además de estimar ciertos parámetros.

5. La simulación puede ser la única posibilidad, debido a la dificultad para realizar experimentos y observar fenómenos en su entorno real - por ejemplo, estudios de vehículos espaciales en sus vuelos interplanetarios.

6. Se requiere la aceleración del tiempo para sistemas o procesos que requieren de largo tiempo para realizarse. La simulación proporciona un control total sobre el tiempo, debido a que un fenómeno se puede acelerar o retardar según se desee. El análisis de los problemas de deterioro urbano corresponde a ésta categoría.

Una ventaja adicional de la simulación radica en su poderosa aplicación educativa y de entrenamiento. El desarrollo y uso de un modelo de simulación le permite al experimentador observar y jugar con el sistema. Esto, a su vez, le ayudará a entender y adquirir experiencia sobre el problema, por lo que auxiliará al proceso de innovación.

Estas razones entre otras han hecho que la simulación sea una de las técnicas cuantitativas de más uso, para resolver problemas de administración. La mayoría de los administradores y analistas están interesados principalmente en obtener una respuesta a sus problemas inmediatos, pero el fin justifica los medios. Es precisamente esta preocupación por los medios la que nos hace preguntarnos si éstos pueden alcanzarse de la manera mas eficiente y efectiva mediante la simulación. Con frecuencia, la respuesta es no, por las siguientes razones:

1. El desarrollo de un buen modelo de simulación es costoso y requiere de mucho tiempo, ya que demanda un alto grado de talento que no se puede encontrar disponible fácilmente.

2. Puede parecer que una simulación refleja con precisión una situación del mundo real cuando, en verdad no lo hace. Varios problemas intrínsecos de la simulación pueden producir resultados erróneos si no se resuelven correctamente.

3. La simulación es imprecisa, y no podemos medir el grado de su imprecisión. El análisis de sensibilidad del modelo para cambiar valores de los parámetros, sólo puede superar parcialmente esta dificultad.

4. Usualmente, los resultados de simulación son numéricos, y dados a cualquier número de puntos decimales que el experimentador seleccione. Por tanto surge el peligro de la "deificación de los números", atribuyéndoles un mayor grado de validez y de precisión del que se puede justificar.

Lo anterior indica que aunque la simulación sea un planteamiento extremadamente valioso y útil para resolver problemas, en realidad no es una panacea para

todos los problema administrativos. El uso y desarrollo de modelos de simulación se refiere en mayor grado a las artes que a las ciencias. En consecuencia, al igual que con otras artes, no es tanto la técnica la que determina el éxito o el fracaso, sino cómo se aplica. Aunque la simulación es un arte, aquellos que poseen la capacidad necesaria de ingenuidad, perspicacia e ingenio pueden dominarla con facilidad.

2.5 LENGUAJES DE PROPOSITO ESPECIAL.

Los primeros esfuerzos en el estudio de simulación se interesaron en definir el sistema que se iba a modelar y en describirlo en términos de diagramas de flujo lógicos y relaciones funcionales. Sin embargo, eventualmente uno se enfrenta al problema de describir el modelo en un lenguaje que sea aceptable para la computadora. La mayoría de las computadoras operan en un lenguaje binario de representación de datos o en algún múltiplo del binario. Debido a que estos lenguajes son inconvenientes para la comunicación, los lenguajes de programación han evolucionado para que los usuarios conversen con la computadora. Lamentablemente se han desarrollado tantos lenguajes de programación de propósito general y de propósito especial a lo largo de los años que es difícil decidir que lenguaje se ajusta mejor o casi se ajusta a una aplicación particular. En consecuencia, el procedimiento usual es adoptar un lenguaje conocido por el analista, no porque sea el mejor sino porque se conoce. Cualquier lenguaje algorítmico general puede expresar el modelo deseado; sin embargo, uno de los lenguajes de simulación especializados puede tener ventajas muy diferentes en términos de facilidad, eficiencia y efectividad.

Las principales diferencias entres los lenguajes de simulación de propósito especial son, por lo general:

1. La organización del tiempo y las actividades
2. La designación y la estructuración de las entidades dentro del modelo.
3. La prueba de actividades y condiciones entre los elementos.
4. Los tipos de pruebas estadísticas que son posibles sobre los datos.
5. La facilidad para cambiar la estructura del modelo.

2.5.1 VENTAJA DE LOS LENGUAJES DE SIMULACION.

El desarrollo evolutivo de los lenguajes de simulación comenzó a finales de los años cincuenta. Al principio, los lenguajes que se usaron en simulación eran lenguajes de una máquina específica o de propósito general. Emshoff y Sisson reconstruyen los antecedentes históricos de los lenguajes de simulación como un proceso en el que los analistas pasan por una secuencia de etapas muy parecida a la siguiente:

a) El analista analiza mentalmente el problema.

b) Puede exponer la situación en algún idioma nativo, tal como el inglés aumentado por una terminología técnica.

c) Describe la situación con tanta precisión como le sea posible en este lenguaje.

d) La descripción es usada por un programador (el cual puede ser el mismo analista) para preparar un programa de cómputo en un lenguaje de propósito general.

Después de repetir esta secuencia muchas veces, los analistas observaron que en gran cantidad de situaciones, siendo simuladas, podían clasificarse ampliamente como sistemas que implicaban el flujo de datos a través de procesos. Debido a que un gran número de los programas tenían procesos funcionalmente similares, la idea de crear lenguajes de propósito especial evolucionó casi simultáneamente dentro de varios grupos de investigadores a principios de los sesenta. Estos lenguajes cambiaron gradualmente de programas de lenguajes ensambladores con características especiales, por medio de lenguajes orientados a problemas comercialmente disponibles, hasta los lenguajes sofisticados de simulación de propósito especial. Cualquier lenguaje de programación algorítmico puede usarse para la modelación de simulación, pero aquellos lenguajes diseñados específicamente para el propósito de simulación por computadora proporcionan ciertas características útiles:

a) Reducen la tarea de programación.

b) Proporcionan una guía conceptual

c) Ayudan a definir las clases de entidades dentro del sistema

d) Proporcionan flexibilidad de cambios.

e) Suministran un medio de diferenciación entre entidades de la misma clase mediante características o propiedades.

f) Describen la relación de las entidades entre si y con su ambiente común.

g) Ajustan el número de entidades conforme varían las condiciones dentro del sistema.

Emshoff y Sisson creen que todas las simulaciones requieren de ciertas funciones comunes que diferencian un lenguaje de simulación de un lenguaje algebraico general o de uno de programas de aplicación comercial o administrativa. Entre éstas se encuentra la necesidad de:

a) Crear números aleatorios.

b) Crear variables aleatorias.

c) Variar el tiempo, ya sea por una unidad fija o hasta que ocurra el siguiente evento.

d) Registrar datos para salida.

e) Efectuar análisis estadísticos sobre datos registrados.

f) Arreglar salidas en formatos específicos.

g) Detectar e informar inconsistencias lógicas y otras condiciones de error.

Además, ellos indican que para las simulaciones en las que se procesan datos discretos mediante operaciones específicas, los siguientes procesos comunes están presentes por añadidura:

a) Determinado el tipo de evento (después de la recuperación de una lista de eventos).

b) Llamar subrutinas para que ajusten las variables de estado, como resultado del evento.

c) Identificar las condiciones de estado específicas.

la lista de eventos y aquellas que representen el estado.

Algunos de los lenguajes de simulación son lenguajes, en el sentido más general, que, además de unir al usuario con la computadora como medio de conversación, le proporcionan ayuda para formular el problema. Al tener un vocabulario y una sintaxis, dichos lenguajes son descriptivos, en consecuencia, los usuarios tienden, después de utilizarlos algún tiempo a pensar en términos de ellos. Por lo tanto, las dos razones mas importantes para utilizar los lenguajes de simulación en lugar de los de propósito general, son la ventaja de la programación y la articulación de conceptos. La articulación de conceptos es importante en la fase de modelado y en el planteamiento en general dedicados a la experimentación del sistema. La utilidad del programa se vuelve importante durante la escritura real del programa de cómputo. Otra ventaja de los lenguajes de simulación es su uso como dispositivos de comunicación o documentación. Cuando están escritos en lenguajes parecidos al inglés las simulaciones pueden explicarse más fácilmente a los gerentes de proyectos y a otros usuarios que no están familiarizados con la programación.

Una desventaja muy importante del uso de los lenguajes de simulación es que, debido a que la mayoría fueron desarrollados por organizaciones particulares para su propio uso y puestos a disposición del público, más como un gesto intelectual y de conveniencia que como un producto comercial, la mayoría de los usuarios, acostumbrados a que los fabricantes de computadoras hagan el trabajo de apoyo de compilación como un servicio, no están dispuestos a realizar este trabajo por si mismos. Sin embargo, cada vez hay más lenguajes nuevos de simulación bien documentados.

En términos generales, la ventaja de usar un lenguaje de propósito especial en lugar de usar uno de propósito general es que aquellas características que son ventajas en uno son desventajas en otro. A continuación se presenta una tabla donde se resumen las ventajas y desventajas de unos y otros lenguajes.

• TABLA 2.1

LENGUAJES DE PROPOSITO GENERAL

VENTAJAS	DESVENTAJAS
1. Número mínimo de restricciones impuestas al formato de salida 2. A menudo muy bien informado en el lenguaje	1. Tiempo de programación más largo. 2. La depuración de los términos del lenguaje de simulación no es una caractrística.

LENGUAJES DE PROPOSITO ESPECIAL

VENTAJAS	DESVENTAJAS
1. Requiere menos tiempo de programación 2. Proporciona técnicas de comprobación de errores superiores a aquellas provistas en los lenguajes de propósito general. 3. Ofrece un medio conciso y directo para expresar los conceptos que surgen en un estudio de simulación. 4. Tiene la habilidad de construir y proporcionar las subrutinas del usuario 5. Genera automáticamente ciertos datos que se necesitan en las corridas de la simulación. 6. Facilita la recopilación y el despliegue de los datos producidos. 7. Controla la administración y la asignación del almacenamiento de la computadora, durante la corrida de la simulación.	1. Debe apegarse a los requerimientos del formato de salida del lenguaje. 2. Flexibilidad reducida en los modelos y tiempo de corrida de computadora incrementado

2.5.2 CLASIFICACION DE LENGUAJES.

En general se puede decir que existen tres técnicas de cómputo para la simulación. Estas son: la digital, la analógica y la híbrida. La figura 2.4 representa nuestro esquema de clasificación. la mayoría de éstos lenguajes tienen varias versiones y dialectos. Por lo tanto, hemos decidido usar nombres de familia o genéricos en lugar de listar todas las diversas versiones.

Las técnicas de simulación digital incluyen el uso de lenguajes de propósito general y de propósito especial. Las ventajas y desventajas relativas a cada una de estas clases ya las presentamos en la tabla 2.1. Sin lugar a dudas, el uso de un lenguaje de propósito general le ofrece al programador la máxima flexibilidad en cuanto al diseño, implantación y uso de su modelo. Sin embargo, esta flexibilidad se obtiene a expensas de mayor dificultad en programación. Gran parte de esta dificultad se deriva de los problemas de ordenamiento, cronometraje y control sin embargo muchos programadores habilidosos y experimentados prefieren los lenguajes de propósito general por la flexibilidad del formato de informe de salida y la habilidad para escribir sus propias subrutinas.

Los lenguajes de simulación digital de propósito especial incluyen dos grupos o tipos diferentes de simulación, los cuales se desarrollaron por separado. Estos grupos (la simulación de procesos de cambio discreto y de cambio continuo) se tratan individualmente, junto con su desarrollo histórico. Hablaremos primero de las técnicas de simulación de cambio continuo. Una técnica incluida en esta categoría representa el final del espectro híbrido, en el cual la hibridación ocurre en un nivel de conceptualización (es decir, la computadora digital imita a la computadora híbrida). La categoría de los lenguajes de simulación de cambio continuo puede dividirse en tres tipos de lenguajes: los lenguajes de simulación analógica; el lenguaje dirigido a la solución de ecuaciones incrementales que representan un sistema cerrado determinístico de lazo, y los lenguajes de simulación de ecuaciones. Los dos primeros tipos son lenguajes orientados a bloques y el tercero se construye directamente a partir de las ecuaciones.

Los lenguajes de simulación de cambio continuo se derivan del trabajo de Selfridge en 1955. Su programa sin nombre ha sido la inspiración para el gran campo de lenguajes de simulación de tipo analógico para las computadoras digitales. Entre los lenguajes de simulación analógica más útiles están MIDAS, PACTOLUS, SCADS, MADBLOC, COBLOC y 1130 CSMP. Estos lenguajes emulan el comportamiento de computadoras analógicas e híbridas sobre una base de componente-por - componente. Por ejemplo, un sumador es reemplazado por un código operacional de suma, un integrador por un código operacional de integración, etc. Todos los lenguajes de simulación analógica se inspiraron y motivaron en el diagrama de bloques analógicos, como un medio simple y conveniente para describir los sistemas continuos.

Tres lenguajes que se ajustan a la representación de un lenguaje útil para modelar variables continuas en valor, pero discretas en tiempo. El DYNAMO (Dynamic Models) se desarrolló en el M. I. T., utilizando ecuaciones incrementadas de primer orden para aproximar el proceso continuo, debido a que son más prácticas que las ecuaciones diferenciales para representar las secuencias de tiempo de la entrada y salida del sistema. Las variables esenciales de un estudio de sistemas dinámicos, las variables de estado y las variables de salida se describen en el DYNAMO mediante ecuaciones de niveles y ecuaciones de velocidad, respectivamente. Las variables de estado (niveles) describen el estado o condición del sistema en un momento determinado; las variables de proporción de flujo (RATE) describen como cambian los estados con el paso del tiempo. Las ecuaciones auxiliares, componentes de las ecuaciones de velocidad, describen totalmente la función de las ecuaciones de velocidad. Aunque el sistema está descrito por las ecuaciones de niveles y velocidad, las ecuaciones auxiliares son muy importantes para representar el control de retroalimentación, debido a que forman la base de control de las velocidades. Si el DYNAMO no está disponible también puede usarse el MIMIC o el CSMP.

Existe una separación en dos escuelas de la primer teoría de simulación de cambio discreto: las introducidas por IBM con su lenguaje GPSS, que usa símbolos de diagrama de flujo como descriptores de modelo básico, y las escuelas orientadas a instrucciones. En general los lenguajes enfocados al diagrama de flujo son más fáciles de aprender, aunque los lenguajes orientados a instrucciones son comúnmente mas flexibles. La mayoría de los lenguajes mas recientes son lenguajes de instrucciones, a pesar de que el lenguaje de diagrama de flujo es atractivo, y aparte del GPSS, ha sido utilizado el SIMCOM y BOSS. En nuestro esquema de clasificación, hemos usado cuatro subcategorías: orientaciones de flujo a actividades, de eventos, de procesos y de transacciones. Los lenguajes orientados a flujo de transacciones son realmente lenguajes de proceso, debido a que toman una perspectiva sinóptica de los sistemas, pero los hemos considerado en una categoría individual debido a su orientación de diagrama de flujo. Los lenguajes de eventos, actividades y procesos (excepto el SIMCOM) usan instrucciones de programación para describir las relaciones de causa y efecto entre los elementos del sistema.

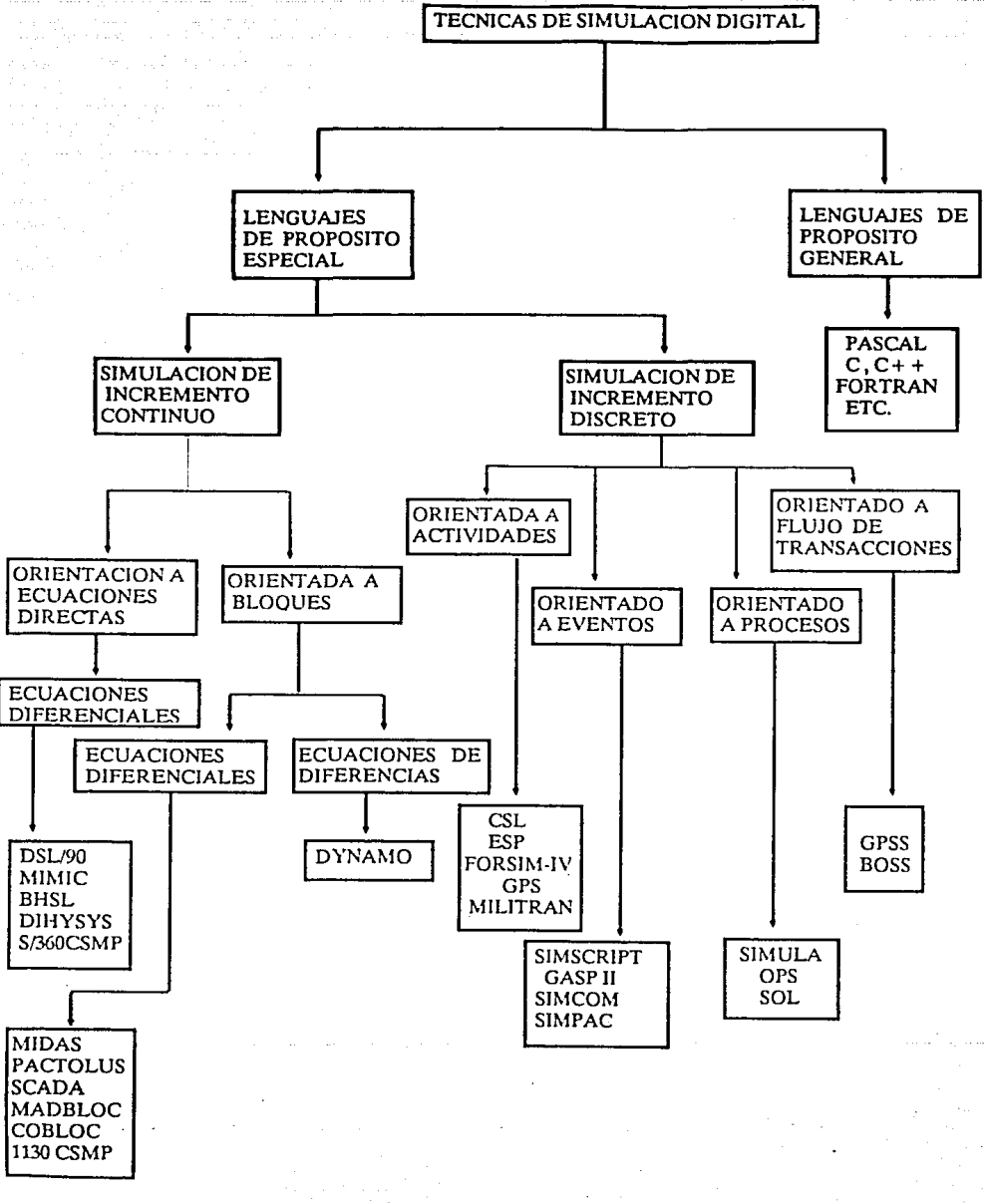
Los lenguajes orientados a actividad representan actos pendientes de tiempo como ocurrencias simultáneas en tiempo simulado. Al usar estos lenguajes, no asignamos un calendario o itinerario a los sucesos dentro de un programa, sino que especificamos las condiciones en las cuales éstas pueden ocurrir. Ninguna instrucción de itinerario de actividad programada aparece en estos lenguajes, pero éstos contienen programas ejecutivos que pueden analizar conjuntos de condiciones, antes de cada incremento de tiempo de simulación, para determinar si cualesquiera de las actividades pueden llevarse a cabo. En este tipo de lenguaje, el programa consiste de una sección de prueba y de una sección de acción. Cuando se adelanta el tiempo de simulación, se examinan todos los programas de actividades para determinar su posible rendimiento. Todas las condiciones de prueba deben cumplirse para que las instrucciones de cambio de estado y de tiempo de calen-

tamiento de la sección de acción se lleven a cabo. Si alguna de las condiciones de prueba no se cumple, las instrucciones de acción se omiten. Mediante una exploración cíclica de los programas con base en actividades, nos aseguramos de que todas las posibilidades tengan oportunidad de llevarse a cabo y de que se responda por todas las interacciones. Entre los lenguajes orientados a actividades están el CSL, ESP, FORSIMIV, GSP y el MILITRAN.

Puede ser que el problema disponible se escriba mas eficientemente en un lenguaje orientado a eventos. Cada evento debe representarse como una ocurrencia instantánea en tiempo simulado, programada para ocurrir cuando la dinámica del modelo perciba que existen las condiciones adecuadas para que ocurra. Los programas separados se requieren, para digamos, un hombre, una máquina y una parte que interactúa. Un programa ejecutivo ordena automáticamente los eventos programados, de tal manera que ocurran de manera adecuada en el tiempo simulado. Entre los lenguajes orientados a eventos se encuentran el SIMSCRPT, GASP, SIMCOM y SIMPAC.

Los lenguajes orientados a procesos intentan combinar la notación concisa de los lenguajes orientados a actividades con las habilidades de los lenguajes orientados a eventos. Un conjunto de eventos asociado a la descripción del comportamiento de un sistema es un proceso. Un proceso existe a través del tiempo y puede tener un comportamiento dinámico. Los procesos son flexibles, pueden programarse para ocurrir, pueden interrumpirse, pueden tener subprocesos que los obedezcan, pueden estar tan programados que son capaces de retrasarse a si mismos y a otros procesos hasta que cumplan ciertas condiciones, etc. Una característica muy importante es que los lenguajes orientados a procesos hacen que un sólo programa actúe como si fueran varios programas, independientemente de que sean controlados por las exploraciones orientadas a actividades o por la formación de itinerarios orientados a eventos. La característica de programación que hace posible este concepto es el punto de reactivación -esencialmente un apuntador que le señala a una rutina de proceso, dónde empezar la ejecución después de que se ha llevado a cabo un comando de demora. El programa ejecutivo tiene más actividad en estos lenguajes que en los lenguajes orientados a actividades u orientados a eventos. E esta categoría se incluyen el SIMULA, OPS, y el SOL.

A pesar de que los diagramas de flujo son a menudo una herramienta vital para describir la lógica y la interrelación de los elementos de los sistemas modelados en los tres tipos de cambio discretos descritos, éstos no son fundamentales para su programación o teoría. En ellos, el usuario construye un modelo de simulación escribiendo instrucciones que definen las condiciones que deben mantenerse para que sucedan ciertas acciones, describen los resultados de dichas acciones y especifican las relaciones de tiempo entre los elementos y actividades del sistema en el cual participan. En el último tipo de lenguaje que se describió, se aplican los conceptos de los lenguajes orientados a procesos, pero los sistemas se modelan mediante la investigación de los flujos de transacciones a través de bloques de


TECNICAS DE SIMULACION DIGITAL
LENGUAJES DE PROPOSITO ESPECIAL
LENGUAJES DE PROPOSITO GENERAL
PASCAL C, C++ FORTRAN ETC.
SIMULACION DE INCREMENTO CONTINUO
SIMULACION DE INCREMENTO DISCRETO
ORIENTACION A ECUACIONES DIRECTAS
ORIENTADA A BLOQUES
ORIENTADA A ACTIVIDADES
ORIENTADO A EVENTOS
ORIENTADO A PROCESOS
ORIENTADO A FLUJO DE TRANSACCIONES
ECUACIONES DIFERENCIALES
ECUACIONES DIFERENCIALES
ECUACIONES DE DIFERENCIAS
DSL/90 MIMIC BHSL DIHYSYS S/360CSMP
DYNAMO
CSL ESP FORSIM-IV GPS MILITRAN
GPSS BOSS
SIMSCRIPT GASP II SIMCOM SIMPAC
SIMULA OPS SOL
MIDAS PACTOLUS SCADA MADBLOC COBLOC 1130 CSMP

actividades. El tiempo de simulación avanza conforme las transacciones pasan a través de los bloques, y las decisiones se toman conforme se reproduce la lógica del sistema simulado. Los lenguajes que se ajustan a esta categoría se conocen como lenguajes de flujo de transacciones. Una persona que conoce los conceptos del diagrama de flujo no tiene problema alguno para aprender a modelar en estos lenguajes. Los bloques especializados se ensamblan para formar estructuras que representan la lógica y el flujo del sistema que se está modelando. El sistema se representa en términos de bloques, el programa crea transacciones, dirigiéndolas a los bloques específicos, y ejecuta las acciones asociadas con cada bloque. Debido a que estos bloques de modelación especializados también son las instrucciones básicas de programación, la construcción de un modelo de diagrama de flujo equivale a elaborar un programa. El precio que se paga por esta facilidad de aprender a usar estos lenguajes es una pérdida de flexibilidad. Los lenguajes en esta categoría son el GPSS y el BOSS.

Cabe mencionar aquí, que a la par del avance de los lenguajes de propósito general, los lenguajes de propósito especial también lo han hecho, y a últimas fechas se tienen paquetes como el MODSIM II que es un lenguaje de programación y simulación orientado a objetos. Nuevas versiones del GPSS, SIMULA, así como nuevos paquetes tales como el MODEL - C que es un programa de simulación con uso de diagramas de bloques o el ACSL que sirve para simular modelos de sistemas de control dinámicos, se encuentran entre los programas comerciales que se difunden en universidades y revistas especializadas. Finalmente, con el desarrollo de la programación orientada a objetos, la cual descansa sobre tres ideas principales, clases, objetos y la herencia de sistemas estructurados, se está desarrollando simulación con el C + +, un nuevo lenguaje orientado a objetos.

2.6 LINEAS DE DESARROLLO.

Hoy en día el desarrollo computacional, tanto en hardware como en software, nos permiten integrar en un modelo de simulación, uno o varios modelos analíticos, criterios de selección de los mismos dependiendo de la situación que prive en el momento en que se requiere su utilización, etc. Los modelos de simulación además de permitirnos modelar situaciones aleatorias, nos permite acercarnos más a lo que se definió como modelo conceptual, que al modelo formal dentro del esquema de modelo diamante.

2.6.1 EN LA MODELACION INTEGRAL DE SISTEMAS.

Le he denominado integral, para resaltar la pretensión de realizar la modelación de sistemas complejos, tanto por el número de componentes como por el número y tipo de relaciones que guarden entre sí. Para tener una idea más exacta de esta pretensión, considere por ejemplo el caso de modelar una institución bancaria: la casa matriz y las diversas sucursales, donde cada una de estas sucursales puede funcionar con características o políticas propias de funcionamiento, con las únicas limitaciones que les puede imponer la casa matriz.

Para realizar un modelo de simulación de este tipo se requiere la utilización de una red de microcomputadoras, que nos permita emular programación en paralelo. La programación en paralelo mediante la utilización de redes de microcomputadoras presupone resolver problemas de sincronización, de los distintos procesos de simulación que simultáneamente se estén realizando.

La realización de este tipo de modelación, requiere trabajo de investigación ya que nos situaría en la frontera tanto de la simulación digital como de la programación en paralelo utilizando micros.

2.6.2 COMO PRODUCTO FINAL.

Elaboración de modelos que simulen el comportamiento de un sistema permitiendo al usuario:

- Evaluar distintas políticas de funcionamiento.
- Cambiar parámetros en las diferentes políticas de funcionamiento.

Este tipo de modelos permiten interaccionar directamente al usuario con el modelo, sin mediar el especialista.

La elaboración de estos modelos implica resolver dos problemas adicionales al originado propiamente por la modelación del sistema y su simulación. El primero de ellos, la captura: selección de política y establecimiento de parámetros. El segundo, la salida: el despliegue de los resultados de la simulación debe ser

accesible al nivel de conocimientos y en la forma deseada por el usuario.

Cualquiera de estas alternativas de desarrollo implica que el producto terminal debe reunir las características de calidad, propias de la profesionalidad de un trabajo de I. de O.

2.6.3 EN LA PROGRAMACION ORIENTADA A OBJETOS.

En la búsqueda continua de mejores propuestas para implantar modelos de simulación se encuentra que existe compatibilidad entre la simulación y los recientes desarrollos en el área de sistemas expertos, en particular parece que la propuesta de programación orientada a objetos de sistemas expertos es apropiada para implantar modelos de simulación.

Existen lenguajes de simulación basados en la programación orientada a objetos. Por ejemplo SIMULA que se ha venido desarrollando en los últimos veinte años, sin embargo no ha alcanzado un uso comercial amplio, esto debido quizá al menos en parte al hecho de que es un lenguaje basado en ALGOL y en muchos casos requiere que se escriban subrutinas en algol para poder simular un sistema completo.

Otro lenguaje orientado a objetos es el C+ + que también se usa para efectuar simulaciones de eventos discretos. Este sistema está basado en la creciente popularidad y eficiencia del lenguaje de programación C. Sin embargo el C+ + parece ofrecer la ventaja de estar mas disponible en la comunidad de la simulación. Además permite que el código de computadora sea modular y reusable lo que desde el punto de vista del desarrollo de la simulación lo hace eficiente.

CAPITULO III

SIMULACION DE UNA SURCUSAL BANCARIA.

Los ordenadores son mejores que nosotros para la aritmética; no es que sean muy buenos, es que nosotros somos bastantes malos para ella.

Isaac Asimov.

INTRODUCCION.

Este sistema se desarrolló para ejemplificar la alternativa de desarrollo en la simulación digital, como producto final. La sucursal simulada es ficticia, los indicadores seleccionados y reportados en la simulación, tal vez no sean los más adecuados, la finalidad del trabajo tiene por objetivo demostrar que podemos desarrollar software de investigación de operaciones con la misma calidad con la que se esta desarrollando en cualquier parte del mundo.

El sistema, resuelve el problema de la captura y modificación de parámetros, garantizando que no salgan de determinados rangos y/o satisfagan determinadas condiciones, por ejemplo: que la suma de algunos parámetros sea 100, no negatividad, etc. La presentación de los resultados se hace mediante gráficas de frecuencias únicamente, esta presentación no excluye la posibilidad, de que en nuevas versiones, se generen otras formas de editar los resultados, como tablas, índices, etc.

PRIMERA PARTE "MANUAL DE USUARIO".

3.1 DEFINICION, OBJETIVOS Y GENERALIDADES DEL SISTEMA.

DEFINICION DE LA PROBLEMATICA A SIMULAR.

El presente sistema simula la atención diaria en cajas de una sucursal bancaria, con las siguientes características y política de servicio.

Clientes. Existe trato preferencial, el modelo permite establecer hasta tres distintos tipos de clientes (Empresariales, Maestra, etc.).

Colas. Colas únicas por tipo de cliente, donde el primero en llegar es primero en ser atendido.

Cajas. Cada cola, tiene asignada un determinado número de cajas.

Operaciones. Cada tipo de cliente puede realizar hasta 10 operaciones distintas (pago de servicios, pago de cheques, etc.), cada una de estas operaciones, por cada tipo de cliente tiene asignadas los siguientes parámetros.

- Tiempo medio de servicio.
- Modificador del tiempo medio de servicio (desviación, etc).
- Distribución de probabilidad del tiempo medio de servicio (uniforme o normal).
- Valor medio.
- Modificador del valor medio.
- Porcentaje de clientes que realizan la operación.

- Nombre de la operación.

Llegadas. Supone llegadas poissonianas, cuyo tiempo medio entre llegadas, puede tener hasta 10 variaciones a lo largo del día.

OBJETIVOS. Al finalizar la simulación, el sistema proporciona mediante gráficas de frecuencias absolutas y acumulada, los resultados obtenidos sobre:

CAJAS.

- Porcentaje de utilización.
- Tiempo máximo de servicio por cliente.
- Tiempo mínimo de servicio por cliente.

CLIENTES.

- Tiempos de cola.
- Tiempos de atención en cajas.

OPERACIONES.

- Porcentaje de operaciones realizadas.
- Utilidad de operación.

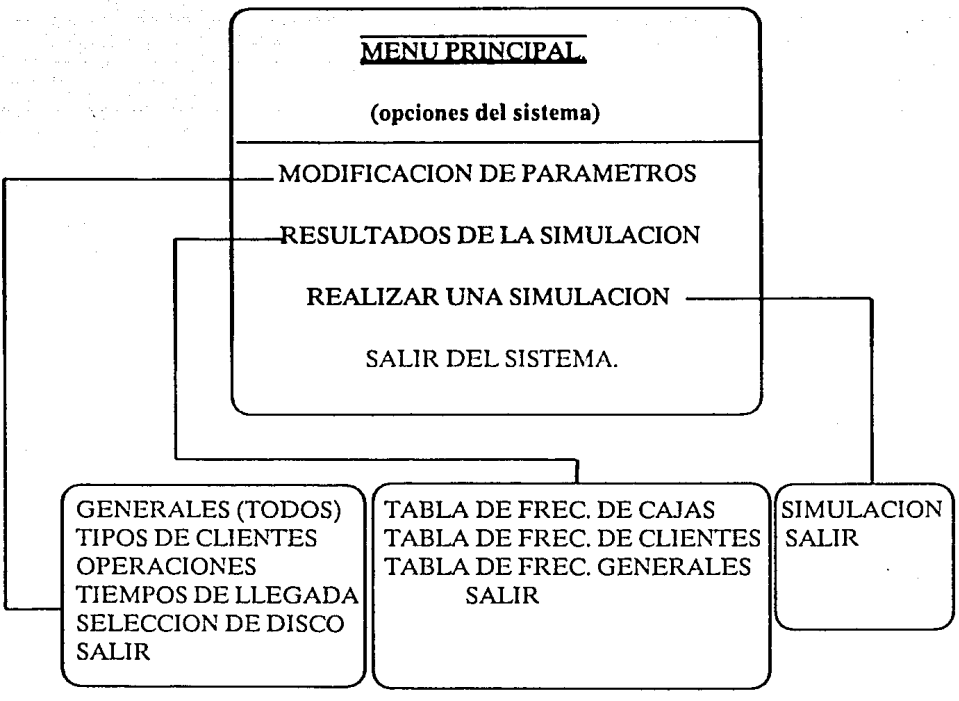
GENERALIDADES.

Las relativamente pocas opciones que ofrece el sistema, determinarón que se optara por un sistema de menús, permitiendo al usuario el fácil acceso a los distintos modulos que integran el sistema. La selección de opciones, mediante el sistema de menús, se hace mediante la utilización de las flechas hacia arriba y hacia abajo, lo que produce en el menú un cambio de coloración en la opción en que se encuentra el usuario, una vez ubicado en la seleccion deseada, se oprime la tecla

3.2 DESCRIPCION MODULAR DEL SISTEMA.

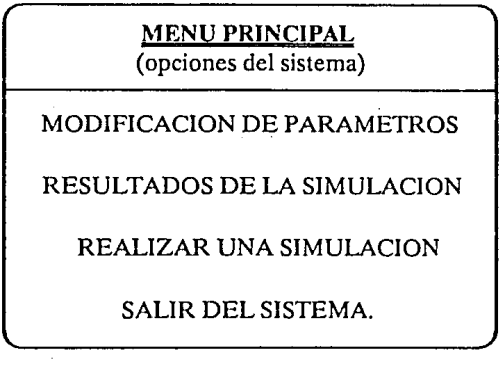
El sistema se compone de tres módulos principales y submodulos consecutivos que conforman a estos. Todos ellos integrados en el modulo principal del sistema, el cual se refleja claramente en el menú principal. Además de estos tres módulos principales, existe un cuarto módulo, **Ayudas**, que en el presente sistema es muy modesto, pero juega un papel importante en cualquier sistema.

El siguiente diagrama, presenta los distintos menus, así como, las relaciones que existen entre los diversos módulos que integran el sistema.



A. MENU PRINCIPAL.

PROPOSITO: Este módulo proporciona al usuario una visión global de las distintas opciones que ofrece el sistema, facilitando a la vez un rápido acceso, a los distintos modulos que conforman el sistema.



DESCRIPCION: Consiste de un menú, con las opciones de: Modificación de parámetros, Resultados de la Simulación, Realizar una simulación y Salir del sistema.

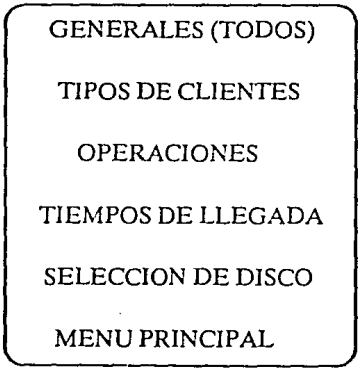
La selección de cualquiera de las opciones enumeradas en la pantalla, lo conducen al módulo que haya sido preferido o bien a abandonar el sistema.

B. OPCION 1. MODULO DE MODIFICACION DE PARAMETROS.

PROPOSITO: Este módulo permite al usuario modificar los parámetros de la simulación a realizar.

El sistema cuenta con un archivo de parámetros, los cuales pueden ser vueltos a llamar y ser utilizados de nuevo o bien modificados, para realizar una nueva simulación.

En forma automática los parámetros de la última simulación realizada son salvados, en caso de que el usuario pase directamente al módulo de simulación, éstos serán los parámetros con los que se realice.



DESCRIPCION:

Consiste de un menú, con las opciones de: Generales (Todos), Tipos de clientes, Operaciones, Tiempos de llegada, Selección de disco, Salir.

La selección de cualquiera de las opciones enumeradas en la pantalla, lo conducen al módulo que haya sido preferido o bien a regresar al menú principal.

B.1 SUBMODULOS DE LA OPCION 1 DEL MODULO DE MODIFICACION DE PARMETROS:

B.1 GENERALES (TODOS LOS PARMETROS).

PROPOSITO: Modificar todos o algunos de los parámetros del sistema.

DESCRIPCION. Mediante una ventana, subtitulada "GENERALES", el sistema solicita, los primeros cuatro parámetros, a partir de los cuales, empezará a solicitar los restantes valores, que requerira la simulación. Los valores solicitados y los límites soportados por el sistema se indican a continuación:

- Número de cajas de la sucursal, el sistema acepta de una a 10 cajas.
- Tipos de cliente. de uno a tres tipos diferentes de clientes y menor que el número total de cajas.
- Número de operaciones diferentes, el sistema soporta de una a diez operaciones distintas por cada tipo de cliente.
- Variaciones de tiempos medios entre llegadas, de uno a diez tiempos medios diferentes.

B1.1 **Asignación de cajas** por tipo de cliente, estos parametros se solicitan solo en el caso de que existan dos o tres tipos de clientes. Las cajas son numeradas del uno al número total de cajas establecidas por el usuario en el paso anterior. Los clientes tipo 1 son atendidos a partir de la caja 1 y se solicita al usuario el número de la última caja que dará atención al cliente tipó uno, en forma automática la siguiente caja se le asigna al cliente tipo dos, de haber tres tipos de clientes se le solicita al usuario el número de la última caja para este tipo de cliente, el sistema en forma automática asigna las restantes cajas para el tipo tres.

B1.2 **Porcentajes de clientes.** Si los tipos de cliente son más de uno, el sistema pide los porcentajes de cada uno de ellos, menos del último, ya que la suma de todos debe ser 100, el sistema verifica que esto se cumpla y en forma automática asigna el porcentaje restante al último tipo de clientes.

B1.3 **Tiempos.** Por cada variacion de tiempos medios entre llegadas, el sistema mediante una ventana de captura solicita:

- Tiempo medio entre llegada.
- Límite superior de validez.

El sistema supone que el limite inferior de validez esta dado por el límite superior del tiempo medio anterior que se recoge, si es el primero su limite inferior esta dado por la hora en que se inicia la simulación.

B1.4 **Operación.** Mediante una nueva ventana se solicita, para cada una de las operaciones indicadas en "generales", el nombre de la operación y por cada tipo de cliente se requiere:

- Tiempo medio de servicio.
- Modificador del tiempo medio de servicio (desviación, etc).
- Distribución de probabilidad del tiempo medio de servicio (uniforme o normal).
- Valor medio.
- Modificador del valor medio.
- Porcentaje de clientes que realizan la operación.

B.2 SUBMODULOS DE LA OPCION 2 DEL MODULO DE MODIFICACION DE PARAMETROS:

B.2 TIPOS DE CLIENTES.

PROPOSITO:Modificar los parámetros actuales, relacionados con el número de tipos de clientes.

DESCRIPCION.Mediante una ventana, solicita al usuario un nuevo número de clientes distintos.

B2.1 **Asignación de cajas** por tipo de cliente, estos parametros se solicitan solo en el caso de que existan dos o tres tipos de clientes. Las cajas son numeradas del uno al número total de cajas establecidas por el usuario en el paso anterior. Los clientes tipo 1 son atendidos a partir de la caja 1 y se solicita al usuario el número de la última caja que dará atención al cliente tipó uno, en forma automática la siguiente caja se le asigna al cliente tipo dos, de haber tres tipos de clientes se le solicita al usuario el número de la última caja para este tipo de cliente, el sistema en forma automática asigna las restantes cajas para el tipo tres.

B2.2 **Porcentajes de clientes.** Si los tipos de cliente son más de uno, el sistema pide los porcentajes de cada uno de ellos, menos del último, ya que la suma de todos debe ser 100, el sistema verifica que esto se cumpla y en forma automática asigna el porcentaje restante al último tipo de clientes.

B.3 SUBMODULOS DE LA OPCION 3 DEL MODULO DE MODIFICACION DE PARAMETROS:

B.3 OPERACIONES.

PROPOSITO:Modificar los parámetros actuales, relacionados con las operaciones.

DESCRIPCION.Mediante una ventana, se solicita al usuario un nuevo número de operaciones distintas a realizar.

B.3.1 Operación. Mediante una nueva ventana se solicita, para cada una de las operaciones indicadas, el nombre de la operación y por cada tipo de cliente del último de los parámetros se requiere:

- Tiempo medio de servicio.
- Modificador del tiempo medio de servicio (desviación, etc).
- Distribución de probabilidad del tiempo medio de servicio. A continuación se indica, las distribuciones que acepta el sistema y el número mediante el cual se accesan:
  - No. de Acceso Distribución
  - 1 Uniforme.
  - 2 Normal.
  - 3 Exponencial.
- Valor medio.
- Modificador del valor medio.
- Porcentaje de clientes que realizan la operación.

B.4 SUBMODULOS DE LA OPCION 4 DEL MODULO DE MODIFICACION DE PARAMETROS:

B.4 TIEMPOS DE LLEGADAS.

PROPOSITO:Modificar los parámetros actuales relacionados con los tiempos medios de llegadas.

DESCRIPCION.Mediante una ventana, se solicita al usuario un nuevo número de variaciones de tiempos de llegada.

B4.3Tiempos. Por cada variacion de tiempos medios entre llegadas, el sistema mediante una ventana de captura solicita:

- Tiempo medio entre llegada.
- Límite superior de validez.

El sistema supone que el limite inferior de validez esta dado por el límite superior del tiempo medio anterior que se recoge, si es el primero su limite inferior esta dado por la hora en que se inicia la simulación.

B.5 SUBMODULOS DE LA OPCION 5 DEL MODULO DE MODIFICACION DE PARAMETROS:

B.5 SELECCION DE DISCO.

PROPOSITO: Seleccionar alguno de los parámetros utilizados en simulaciones anteriores y que se encuentran almacenados.

DESCRIPCION: Por orden de antigüedad, de más a menos, el sistema lee los parámetros almacenados y los presenta en la pantalla. Despúes de presentar cada registro, aparece una ventana preguntando, si continuo el proceso o si selecciona el actual.

B.6 SUBMODULOS DE LA OPCION 6 DEL MODULO DE MODIFICACION DE PARAMETROS:

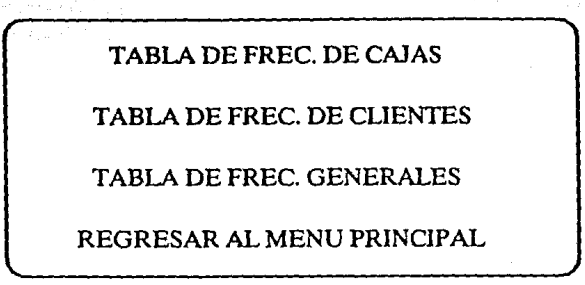
B.5 REGRESAR AL MENU PRINCIPAL.

PROPOSITO: Permite al usuario regresar al menú principal.

DESCRIPCION: Regresa al usuario al menú principal.

C. OPCION 2. MODULO DE RESULTADOS DE LA SIMULACION.

PROPOSITO: Este modulo permite al usuario ver los resultados gráficos de la última simulación realizada.



DESCRIPCION: Consiste de un menú, con las opciones de: Tabla de frecuencias de cajas, de clientes, generales y regresar al menú principal.

La selección de cualquiera de las opciones enumeradas en la pantalla, lo conducen al módulo que haya sido preferido.

C. SUBMODULOS DE LA OPCION 1 DEL MODULO DE RESULTADOS DE LA SIMULACION.

C.1 TABLAS DE FRECUENCIAS DE CAJAS

PROPOSITO: Presentar las graficas de frecuencias absolutas y la acumulada de frecuencias relacionadas con las cajas.

DESCRIPCION: Las frecuencias presentadas son: porcentajes de utilización, tiempos máximos de servicio por cliente y tiempos minimos de atención en cajas. Las tablas anteriores por cada tipo de clientes. Como se ilustra en la figura.

C.2 TABLAS DE FRECUENCIAS DE CLIENTES

PROPOSITO: Presentar las graficas de frecuencias absolutas y la acumulada de frecuencias relacionadas con los clientes.

DESCRIPCION: Las frecuencias presentadas son: porcentajes de tiempos de espera, tiempos de atención en cajas. Las tablas anteriores por cada tipo de cliente.

C.3 TABLAS DE FRECUENCIAS GENERALES.

PROPOSITO: Presentar las graficas de frecuencias absolutas y la acumulada de frecuencias relacionadas con los indicadores del sistema.

DESCRIPCION: Las frecuencias presentadas son: porcentajes de tiempos de cola, número total de clientes atendidos. Las tablas anteriores por cada tipo de cliente.

C.4 REGRESAR AL MENU PRINCIPAL.

PROPOSITO: Permite al usuario regresar al menú principal.

DESCRIPCION: Regresa al usuario al menú principal.

D. OPCION 2. MODULO DE REALIZAR UNA SIMULACION.

PROPOSITO. Realizar la simulación un determinado número de veces.

DESCRIPCION: Aparece una pantalla solicitando al usuario, el número de veces que solicita se realice la simulación, este número puede variar entre 1 y 100, el sistema establece por ausencia el número de 30. La simulación se realiza con los parametros seleccionados por el usuario en la opción 1, o bien, si el usuario, no hizo ninguna selección de parámetros, el sistema toma los de la última simulación realizada.

D.1 REGRESAR AL MENU PRINCIPAL.

PROPOSITO: Permite al usuario regresar al menú principal.

DESCRIPCION: Regresa al usuario al menú principal.

SEGUNDA PARTE.

MANUAL TECNICO DEL SISTEMA.

INTRODUCCION.

El sistema se elaboró en el lenguaje de proposito general PASCAL versión 5.5. y haciendo uso de las utilerias con que cuenta dicho lenguaje.

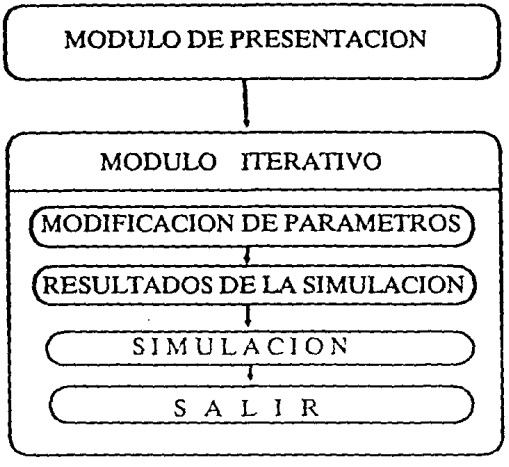
El sistema esta conformado por cuatro unidades, denominadas "PARA2", "SIMULA2", "ESTADIS" Y "TYPOTESIS". Las tres primeras unidades, conforman los modulos principales que se integran en el programa principal denominado "BANCOSIM". En la unidad "TYPOTESIS", se establecierón los tipos y variables que se utilizan en los distintos módulos.

Este manual se conforma de 6 partes. En la primera se hace una descripción modular detallada del sistema, las restantes partes corresponden a los programas fuentes (comentados) de las cuatro unidades y al programa principal, el orden en que se presentan es: simula2, para2, estadis, typostesis, bancosim. Cada uno de estos apartados inicia con algun comentario, a continuación se presentan los programas fuentes del sistema. Los textos que se encuentran entre lineas de llaves ({{{}}}), son comentarios que se considera ayudara al lector a comprender los listados.

Sobra decir que se requieren conocimientos mínimos de pascal para abordar este manual.

DESCRIPCION MODULAR DEL SISTEMA.

El sistema se conforma de dos modulos, un modulo de presentación del sistema y un modulo iterativo, donde el usuario puede moverse libremente entre las distintas opciones que ofrece el sistema, el siguiente diagrama ilustra esta integración.



SUBMODULOS PRINCIPALES DEL MODULO ITERATIVO.

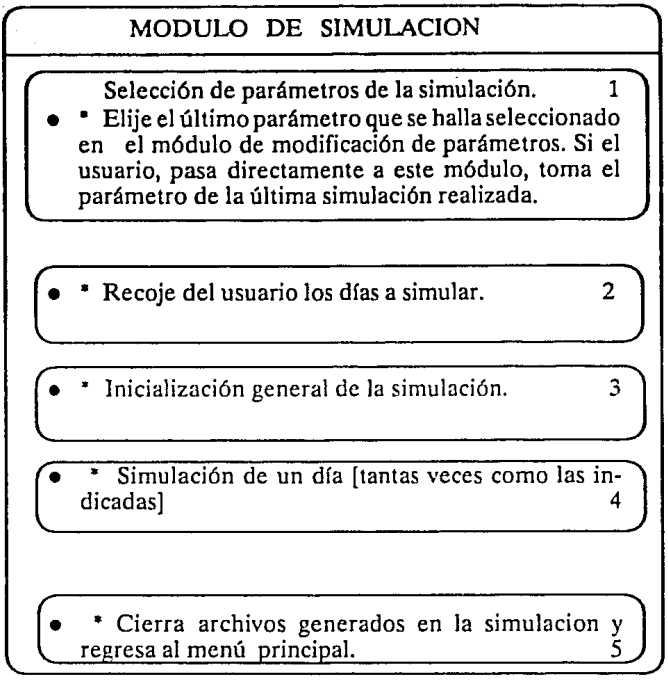
MODIFICACION DE PARAMETROS.

Este módulo conforma la unidad "PARA2", integrada por procedimientos de lectura de los parámetros, verificando que dichos valores se encuentren dentro de determinados rangos. Los procedimientos que constituyen esta unidad podemos clasificarlos en procedimientos de apoyo, para facilitar la captura de datos, de presentación de datos y de captura de datos. Lo que permite que los de presentación de datos puedan utilizarse indistintamente tanto para la captura de información, como para la simple presentación de valores, según sea requerido en el sistema.

RESULTADOS DE LA SIMULACION.

Este módulo constituye la unidad denominada "ESTADIS", constituida por procedimientos de: a) lectura y levantamiento de tablas de frecuencias de archivos de datos generados en una corrida de simulación y b) procedimientos de graficación, donde se grafican las estadisticas levantadas en los procedimientos tipo a).

SIMULACION. El módulo objetivo del sistema, que integra la unidad "SIMULA2", para su mejor comprención a continuación damos una descripción de los submodulos que lo conforman.



Estos son los submódulos, en que se concebio el módulo de simulación y se refleja en el último procedimiento de la unidad "SIMULA2", denominado corrida. Los primeros dos submódulos consisten de una serie de instrucciones dadas en ese mismo procedimiento, donde el primero verifica si el usuario a seleccionado algún parametro, en caso afirmativo continua al siguiente módulo, si el resultado es negativo abre un archivo donde esta almacenado el último parámetro utilizado en una corrida y lo selecciona. El segundo submodulo consiste en un procedimiento de lectura, donde verífica que el valor este entre 1 y 100. Los siguientes submodulos los podemos descomponer en los siguientes submodulos:

DESCRIPCION DE LOS SUBMODULOS PRINCIPALES	NOMBRE DE LOS PRO-CEDIMIENTOS.
INICIALIZACION GENERAL DE LA SIMULACION. * Crea las colas de los clientes. * Crea Archivos de estadisticas. * Crea lista de esventos futuros.	Inicia1; Crea_Colas. Crea_Arch_Est. instrucciones en el mismo procedimiento.
SIMULACION DE UN DIA * Inicializa contadores de estadistica diaria, variables de control y pantalla	Inicia2. Enuncia_cola.
* Modulo iterativo. * Selecciona el siguiente evento actualiza el t_reloj, decrementa el contador de eventos futuros, ejecuta el evento a realizar, el cual puede ser:	Siguiente_evento.
a) llegada de cliente. / b) liberación de caja.	a) nacimiento. b) libera_caja.
* Levanta estadisticas diarias.	Levanta_est_caja.

UNIDAD SIMULA2.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}
Este es el modulo de simulación del sistema. La mayoria de los procedimientos vienen comentados en el programa fuente, solo se comentan aquellos que no esten comentados y no se hallan mencionado explicitamente en el apartado anterior.
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}

```
UNIT SIMULA2;
{***********************************************************}
{***********************************************************}
{           MODULO DE SIMULACION                }
{***********************************************************}
INTERFACE
USES Dos, TPCRT, TPSTRING, TPWINDOW, TPSCREEN, SOUNDS,
    ventanas, GRAPH,para2, typotesi;
var
   E_caj3,
   E_caj4          : array [1..3] of Caja_est;
   llegada         : word;
   cab_evento,
   fin_evento       : evento;
   arch_col         : file of aoperentero;
   tiempo_total_cola,
   clientes_cola_cero  : aoperentero;
{***********************************************************}
{***********************************************************}
{         DECLARACION DE PROCEDIMIENTOS            }
{***********************************************************}
procedure Crea_Arch_Est;
procedure cierra_archivos;
procedure levanta_estadistica(var aux5: cliente);
procedure Levanta_est_caja;
procedure nace_cliente(var aux5 : cliente);
procedure muere_cliente(var aux5 : cliente);
procedure nace_evento(var eve:evento);
procedure muere_evento (var eve:evento);
procedure escribe_caja(var ind:byte);
Procedure Enuncia_cola ;
Procedure Escribe_cola (var ind: byte);
procedure iserta (var eve:evento);
function uniforme (media,mas_menos:real):integer;
function normal(media,desv:real):integer;
```

```
procedure siguiente_llegada(var st:byte;var siguiente,t_reloj:word);
function checa_caja(var nd : byte):byte;
procedure forma_cola(var aux5:cliente);
procedure crea_colas;
procedure saca_cola(var aux5:cliente;var h: byte);
procedure llena_cliente (var aux5:cliente);
procedure ocupa_caja (var ind :byte; var aux5:cliente);
procedure Libera_caja(var ind:byte; var aux5:cliente);
procedure crea_cajas;
procedure inicia2;
procedure inicia1;
procedure nacimiento ;
procedure siguiente_evento(qu :integer);
procedure corrida;
{**********************************************************}
 IMPLEMENTATION
{**********************************************************}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Este procedimiento además de la creación de los archivos de estadisticas, se inicializan contadores que solo requieren inicializarce una sola ocación, durante la corrida de simulación. El siguiente procedimiento cierra los archivos de estadis-. ticas, una vez concluida la simulación.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
{**********************************************************}
procedure Crea_Arch_Est;
var s : string;
  i : integer;
begin
  assign(F_Ca,'cajas.dat');
  rewrite(F_Ca);
  assign(arch_clientes,'clientes.dat');
  assign(arch_No_oper,'No_oper.dat');
  assign(arch_colas,'colas.dat');
  rewrite (arch_colas);
  rewrite (arch_No_oper);
  rewrite (arch_clientes);
  assign(arch_col,'prom_col.dat');
  rewrite(arch_col);
  for i:= 1 to parametro.nume_tipos_clientes do
   begin
    str(i,s);
    assign(F_C[i],'clien'+ s+'.dat');
```

```
    rewrite (F_C[i]);
    with Ma_clie[i] do
     begin
      tipo : = 0;
      tiempo_atencion: = 0;
      tiempo_de_cola : = 0;
     end;
    with Mi_clie[i] do
     begin
      tipo : = 0;
      tiempo_atencion: = 10000;
      tiempo_de_cola : = 10000;
     end;
 with E_caj3[i] do
 begin
   numero: = 0;
   t_max: = 0;
   T_min: = 0;
   t_ocupa: = 0;
   cliente_atendidos: = 0;
 end;
 with E_caj4[i] do
 begin
   numero: = 11;
   t_max: = 10000;
   T_min: = 10000;
   t_ocupa: = 2;
   cliente_atendidos: = 1000;
 end;
end;
 for i: = 1 to parametro.nume_de_operaciones do
 begin
    str(i,s);
    assign(F_O[i],'oper' + s + '.dat');
    rewrite(F_O[i]);
    with Ma_e[i] do
    begin
     tipo: = i;
     tiempo: = 0;
     valor: = 0;
```

```
      end;
      with Mi_e[i] do
      begin
        tipo:=i;
        tiempo:=10000;
        valor:=10000;
      end;
    end;
end;
{*************************************************************}
procedure cierra_archivos;
var   i : integer;
begin
   for i:=1 to  parametro.nume_tipos_clientes do
    begin
     write(F_Ca,E_caj3[i]);
     write(F_Ca,E_caj4[i]);
    write(F_C[i],Ma_clie[i]);
     write(F_C[i],Mi_clie[i]);
     close (F_C[i]);
    end;
   close(F_Ca);
   close(arch_clientes);
   close(arch_No_oper);
   close(arch_colas);
   close(arch_col);
   for i:=1 to parametro.nume_de_operaciones do
   begin
     write(F_O[i],Ma_e[i]);
     write(F_O[i],Mi_e[i]);
     close (F_O[i]);
   end;
 end;
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Los siguientes procedimientos, levantan las estadisticas de la simulación, el primero lo realiza por cada cliente y el siguiente las correspondientes a cada día simulado, cajas, colas máximas, etc.

```
 {*************************************************************}
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
 {*************************************************************}
```

```
procedure levanta_estadistica {(var aux5: cliente)};
var
qu,i:byte;
begin
  with aux5^ do
  begin
     for i:=1 to parametro.nume_de_operaciones do
      begin
       if operacion[i].tiempo0 then begin
        E_oper.tipo:=tipo;
        inc(cuen_oper[tipo,i]);
        E_oper.tiempo:=operacion[i].tiempo;
        E_oper.valor:=operacion[i].valor;
        write(F_O[i],E_oper);
       if Ma_e[i].tiempo^.tiempo then  Ma_e[i].tiempo:=E_oper.tiempo;
       if Ma_e[i].valor^.valor then  Ma_e[i].valor:=E_oper.valor;
        if Mi_e[i].tiempoE_oper.tiempo then  Mi_e[i].tiempo:=E_oper.tiempo;
        if Mi_e[i].valorE_oper.valor then  Mi_e[i].valor:=E_oper.valor;
        end{del then}
      end;{del for}
      cuenta_cliente[tipo]:=cuenta_cliente[tipo]+1;
      E_clie.tipo:=tipo;
      E_clie.tiempo_atencion:=despacho;
      E_clie.tiempo_de_cola:=t_reloj-nacimiento;
      tiempo_total_cola[tipo]:= tiempo_total_cola[tipo]+
                    ((t_reloj-nacimiento)/60);
  clientes_cola_cero[tipo]:=clientes_cola_cero[tipo]+ despacho/60;
      write(F_C[tipo],E_clie);
          if Ma_clie[tipo].tiempo_atencion.< tiempo_atencion then
Ma_clie[tipo].tiempo_atencion:=E_clie.tiempo_atencion;
      if Ma_clie[tipo].tiempo_de_cola < tiempo_de_cola
      then Ma_clie[tipo].tiempo_de_cola:=E_clie.tiempo_de_cola;
          if Mi_clie[tipo].tiempo_atencionE_clie.tiempo_atencion  then
Mi_clie[tipo].tiempo_atencion:=E_clie.tiempo_atencion;
      if Mi_clie[tipo].tiempo_de_colaE_clie.tiempo_de_cola
      then Mi_clie[tipo].tiempo_de_cola:=E_clie.tiempo_de_cola;
   end;{del with}
 end;
 {*********************************************************}
 procedure Levanta_est_caja;
 var
 j, i : integer;
```

```
begin
 for i: = 1 to parametro.nume_de_cajas do
 begin
  with parametro do
   if i < = ambio_de_caja[1] then j: = 1 else
   if i < = cambio_de_caja[2] then j: = 2 else j: = 3;
  with E_caja do
  begin
    numero: = i;
    t_max: = caja[i].t_max_de_atencion;
    T_min: = caja[i].t_minimo_de_atencion;
    t_ocupa: = caja[i].t_de_ocupacion/t_reloj;
    cliente_atendidos: = caja[i].clientes_atendidos;
    if E_caj3[j].t_max < t_max then E_caj3[j].t_max: = t_max;
    if E_caj3[j].t_min < t_min then E_caj3[j].t_min: = t_min;
    if E_caj3[j].t_ocupa < t_ocupa then E_caj3[j].t_ocupa: = t_ocupa;
    if E_caj3[j].cliente_atendidos < cliente_atendidos
     then E_caj3[j].cliente_atendidos: = cliente_atendidos;
    if E_caj4[j].t_max > t_max then E_caj4[j].t_max: = t_max;
    if E_caj4[j].t_min > t_min then E_caj4[j].t_min: = t_min;
    if E_caj4[j].t_ocupa : = t_ocupa then E_caj4[j].t_ocupa: = t_ocupa;
    if E_caj4[j].cliente_atendidos > cliente_atendidos
     then E_caj4[j].cliente_atendidos: = cliente_atendidos;
  end;{del with}
  write(F_Ca,E_caja);
 end;{del for}
 write(arch_clientes,cuenta_cliente);
 write(arch_No_oper,cuen_oper);
 write(arch_colas,cola1);
 for i: = 1 to parametro.nume_tipos_clientes do
  begin
   tiempo_total_cola[i]: = tiempo_total_cola[i]/cuenta_cliente[i];
   clientes_cola_cero[i]: = clientes_cola_cero[i]/cuenta_cliente[i];
  end;
 write(arch_col,tiempo_total_cola);
 write(arch_col,clientes_cola_cero);
end;
{*********************************************************}
procedure nace_cliente {(var aux5 : cliente)};
begin
 if cola_muertos^.sig = cabeza_muertos then new(aux5)
```

```
 else begin
      aux5:=cola_muertos^.sig;
      cola_muertos^.sig:=aux5^.sig
     end;
end;
{*************************************************************}
procedure muere_cliente {(var aux5 : cliente)};
begin
 aux5^.sig:=cola_muertos^.sig;
 cola_muertos^.sig:=aux5;
end;
{*************************************************************}
procedure nace_evento {(var eve:evento)};
begin
  if cola_evento^.siguiente= cabeza_evento then new(eve)
  else
     begin
      eve:=cola_evento^.siguiente;
      cola_evento^.siguiente:=eve^.siguiente;
     end;
end;
{*************************************************************}
procedure muere_evento {(var eve:evento)};
begin
  eve^.siguiente:=cola_evento^.siguiente;
  cola_evento^.siguiente:=eve;
end;
procedure escribe_caja {(var ind:byte)};
var
  cont  : byte;
  aux   : real;
begin
   cont:=3+8*(ind-1); aux:=caja[ind].t_de_liberar;
   if caja[ind].desocupada then mensaje(cont,6,'LIBRE',7) else
     Escribe_real(cont,6,aux,5,0,7);
   escribe_entero(cont-1,7,caja[ind].clientes_atendidos,6,7);
end;
{*************************************************************}
{*************************************************************}
 Procedure Enuncia_cola ;
var
```

```
  cont,d,e   : byte;
  aux        : real;
begin
 for d: = 1 to parametro.nume_de_cajas do
  begin
   cont: = 3 + 8*(d-1);
   mensaje(cont,5,'CAJA',d);
   with caja[d] do
   begin
      if desocupada then
      mensaje(cont,6,'LIBRE',7);
      aux: = t_de_liberar;
      escribe_real(cont,6,aux,5,0,11);
   end;
  end;
 for d : = 1 to parametro.nume_tipos_clientes do
  begin
   cont: = 20*d;
   mensaje(cont-1,10,' FILA ',7);
   mensaje(cont,11,'VACIA',7)
  end;
  mensaje (35,19,'EVENTOS FUTUROS',8);
end; { Enuncia_cola}
{*************************************************************}
{*************************************************************}
Procedure Escribe_cola {( ind: byte)};
var
  cont   : byte;
  aux    : real;
begin
    cont: = 20*ind;
    with cabeza[ind]^ do
    begin
      aux: = despacho;
      Escribe_real(cont,11,aux,5,0,7);
    end;
end; {Procedure Escribe_cola (var ind: byte)}
{*************************************************************}
{ iserta en la lista de eventos futuros}
{*************************************************************}
procedure iserta {(var eve:evento)};
```

```
var
 even4, even5 : evento;
begin
   even4: = fin_evento;
   repeat
     even5: = even4 ^ .siguiente;
     if even5 ^ .tiempo  eve ^ .tiempo then
     begin
        eve ^ .siguiente: = even5;
        even4 ^ .siguiente: = eve;
     end {del then}
     else
     even4: = even5;
   until even4 ^ .siguiente = eve;
    ef: = ef + 1;
    escribe_entero(52,19,ef,5,13);
    mensaje (60,19,'tipo de evento',8);
    Escribe_byte(65,20,eve ^ .tipo,5,7);
end; { iserta (var eve:evento)}
{*************************************************************}
{ genera nmeros distribuidos uniformemente  en un
 intervalo de longitud 2 * mas_menos y de media media.  }
{*************************************************************}
function uniforme {(media,mas_menos:real):integer};
var
 aux  : real;
begin
   aux: =  media-mas_menos;
   uniforme: = trunc(aux  +  random * (2*mas_menos + 1));
end;
{*************************************************************}
{ genera nmeros con distribucin normal de par metros
 media,desv. no permite salidas negativas          }
{*************************************************************}
function normal {(media,desv:real):integer};
var
aux   : real;
i     : byte;
begin
   aux: = 0;
   for I: = 1 to 12 do
    aux: = aux +  random;
```

```
    aux:=(aux-6)*desv;
    normal:=abs(trunc(media+aux));
end;

{*************************************************************}
{ genera el tiempo de llegada del siguiente cliente y
 actualiza el tiempo medio de llegada                }
{*************************************************************}
 procedure siguiente_llegada{(var st:byte;var siguiente,t_reloj:word)};
var
  aux  : real;
begin
 with parametro do
   begin
    if t_de_llegadas[st].hasta < t_reloj then st:=st+1;
    aux := -(60/t_de_llegadas[st].c_p_m)*ln(random);
    siguiente:=t_reloj+trunc(aux);
   end;
end;{de siguiente_llegada}
 {*************************************************************}
{ proporciona el nmero de la ltima caja desocupada que
 atiende la cola número ind, regresa cero si estan todas ocupadas }
{*************************************************************}
function checa_caja {( nd : byte):byte};
var
aux,i,j  : byte;
begin
 with parametro do
  begin
    checa_caja:=0;
    if nd=1 then aux:=1 else aux:=cambio_de_caja[nd-1]+1;
    if nd=3 then j:=parametro.nume_de_cajas
      else j:= cambio_de_caja[nd];
    for i:=aux to j do
     if caja[i].desocupada=true then checa_caja:=i;
  end;{del with}
end;
 {*************************************************************}
{forma al cliente aux5, en la cola que le corresponde }
{*************************************************************}
procedure forma_cola {(var aux5:cliente)};
var h  : byte;
```

```
begin
  h:=aux5^.tipo;
  aux5^.sig:=cola[h]^.sig;
  cola[h]^.sig:=aux5;
  inc(cabeza[h]^.nacimiento);
  inc(cabeza[h]^.despacho);
  if cola1[h] cabeza[h]^.despacho then
   cola1[h]:= cabeza[h]^.despacho;
  escribe_cola(h);
end; { forma_cola(var aux5:cliente)}
{**********************************************************}
 { creacion de las colas }
{**********************************************************}
procedure crea_colas;
var
 i : byte;
begin
  { Creando colas por tipo de clientes }
  for i:=1 to parametro.nume_tipos_clientes do
  begin
     new(cola[i]);
     new(cabeza[i]);
     with cabeza[i]^ do
     begin
       tipo:=i;
       nacimiento:=0;
       despacho:=0;
       salida:=0;
       sig:=nil;
     end; {del with}
     cola[i]^.sig:=cabeza[i];
  end; {del for}
  {Creando colas de muertos}
  new(cola_muertos);
  new(cabeza_muertos);
  with cola_muertos^ do
   begin
    sig:=cabeza_muertos;
    tipo:=0;
    nacimiento:=0;
    despacho:=0;
    salida:=0;
```

```
  end;
  with cabeza_muertos ^ do
  begin
    sig: = nil;
    tipo: = 0;
    nacimiento: = 0;
    despacho: = 0;
    salida: = 0;
  end;
end;{crea_colas}
{*********************************************************}
 {saca de una cola al primero de la fila}
{*********************************************************}
procedure saca_cola {(var aux5:cliente;var h: byte)};
var
 aux4  : cliente;
begin
  aux4: = cola[h];
  aux5: = aux4 ^ .sig;
  while aux5 ^ .sigcabeza[h] do
   begin
     aux4: = aux5;
     aux5: = aux5 ^ .sig;
   end;{del while}
  aux4 ^ .sig: = cabeza[h];
  aux5 ^ .sig: = nil;
  dec(cabeza[h] ^ .despacho);
  inc(cabeza[h] ^ .salida);
  escribe_cola(h);
end; {procedure saca_cola(var aux5:cliente;var h: byte);}
{*********************************************************}
{ LLena los registros de un nuevo cliente            }
{*********************************************************}
procedure llena_cliente {(var aux5:cliente)};
var
  y   : real;
  i,z   : integer;
begin
   y: = random; z: = 0;
   with parametro do
   begin
     case nume_tipos_clientes of
```

```
        1 : aux5^.tipo:=1;
        2 : if y por_cliente[1]/100 then aux5^.tipo:=1
            else aux5^.tipo:=2;
        3 : if y por_cliente[1]/100 then aux5^.tipo:=1 else
            if y por_cliente[1]/100+por_cliente[2]/100
            then aux5^.tipo :=2 else aux5^.tipo:=3;
       end; { del case }
      repeat
       for i:=1 to nume_de_operaciones do
       begin
        y:=random;
        if y<operacion[i].porcentaje[aux5^.tipo]/100 then
         begin
          with operacion[i] do
           begin
            if dis_t_m_s[aux5^.tipo] 2 then
          aux5^.operacion[i].tiempo:=uniforme(t_m_s[aux5^.tipo],
m_t_m[aux5^.tipo])
            else
  aux5^.opeacion[i].tiempo:=normal(t_m_s[aux5^.tipo],m_t_m[aux5^.tipo]);
            if dis_v_m[aux5^.tipo] <5 then
     aux5^.operacion[i].valor:=uniforme(valor_medio[aux5^.tipo],
dis_v_m[aux5^.tipo])
                                  else  aux5^.operacion[i].valor:=nor-
mal(valor_medio[aux5^.tipo],dis_v_m[aux5^.tipo]);
           end;{with de operacion}
         end { then}
          else
           begin
              aux5^.operacion[i].tiempo:=0;
              aux5^.operacion[i].valor:=0;
           end; {else}
           z:=z+aux5^.operacion[i].tiempo;
       end; { del for}
      until z>0;
     end; {de with de parametro}
     with aux5^ do
     begin
      nacimiento:=t_reloj;
      despacho:=z; {tiempo de atencion en cajas}
      mensaje (3,20,'t_nacimiento',8);
```

```
    mensaje (3,21,'tiempo de servicio',8);
    mensaje (3,22,'tipo de cliente',8);
    y:=nacimiento;
    escribe_real(23,20,y,5,0,9);
    y:=despacho;
    escribe_real(23,21,y,5,0,9);
    escribe_byte(23,22,tipo,5,9);
   end;
end;{de llena_cliente}
{*********************************************************}
{ un cliente pasa a ser atendido en la caja numero ind    }
{*********************************************************}
procedure ocupa_caja {( ind :byte; var aux5:cliente)};
var
 t  : real;
begin
  t:=aux5^.despacho+t_reloj;
  aux5^.salida:=trunc(t);
  with caja[ind] do
  begin
   desocupada:=false;
   t_de_ocupacion:=t_de_ocupacion+aux5^.despacho;
   t_de_liberar:= trunc(t){t_reloj+aux5^.despacho};
  mensaje (50,21,'caja t_de_liberar ',8);
  write(t_de_liberar);
   if t_max_de_atencion aux5^.despacho
     then t_max_de_atencion:=aux5^.despacho;
   if t_minimo_de_atencionaux5^.despacho
     then t_minimo_de_atencion:=aux5^.despacho;
   clientes_atendidos:=clientes_atendidos+1;
   escribe_caja(ind);
  end;{del with de caja}
  levanta_estadistica(aux5);
  muere_cliente(aux5);
  nace_evento(nuevo_evento);
  nuevo_evento^.tiempo:=caja[ind].t_de_liberar;
  nuevo_evento^.tipo:=ind;
  iserta(nuevo_evento);
end;{ocupa_caja}
{*********************************************************}
{ libera caja, si la cola que le corresponde es no vacia,
```

```
 saca al primero y ocupa de nuevo la caja          }
{*************************************************************}
procedure Libera_caja {(var ind:byte; var aux5:cliente)};
var j : byte;
begin
    caja[ind].desocupada: = true;
    if ind < = parametro.cambio_de_caja[1] then j : = 1
     else if in < = parametro.cambio_de_caja[2] then j : = 2
      else j: = 3;
    if cola[j] ^ .sigcabeza[j] then
      begin
         saca_cola(aux5,j);
         ocupa_caja(ind,aux5);
      end;
    escribe_caja(ind);
end;
{*************************************************************}
{*************************************************************}
procedure crea_cajas;
var
i  : byte;
begin
   for i: = 1 to parametro.nume_de_cajas do
   begin
    with caja[i] do
     begin
      desocupada: = true;
      t_de_ocupacion: = 0;
      t_de_liberar: = 0;
      t_max_de_atencion: = 0;
      t_minimo_de_atencion: = 5000;
      clientes_atendidos: = 0;
     end;{del with}
   end;{del for}
end;{crea_cajas}
{*************************************************************}
{ inicializo colas, cajas, t_reloj, controlador de tiempos
 st y se crea la lista de eventos futuros          }
{*************************************************************}
procedure inicia2;
var
 aux : real;
```

```
 i,j  : integer;
begin
  st: = 1;
  for i: = 1 to 3 do
  begin
   cuenta_cliente[i]: = 0;
   cola1[i]: = 0;
   tiempo_total_cola[i]: = 0;
   clientes_cola_cero[i]: = 0;
  end;
  for j: = 1 to 3 do
  for i: = 1 to num_de_oper do
  cuen_oper[j,i]: = 0;
  ef: = 0;
  t_reloj: = 1;
  crea_cajas;
  siguiente_llegada(st,llegada,t_reloj);
  nace_evento(nuevo_evento);
       aux: = llegada;
  nuevo_evento ^.tiempo: = llegada;
  nuevo_evento ^.tipo: = 0;
  iserta(nuevo_evento);
end;
{**********************************************************}
procedure inicia1;
var
 aux  : real;
 i    : integer;
begin
  crea_colas;
  Crea_Arch_Est;
{  randomize;}
  new(cab_evento);
  new(fin_evento);
  cab_evento ^.siguiente: = nil;
  fin_evento ^.siguiente: = cab_evento;
  cab_evento ^.tiempo: = 0;
  fin_evento ^.tiempo: = 32000;
  new(cola_evento);
  new(cabeza_evento);
  cola_evento ^.siguiente: = cabeza_evento;
  cabeza_evento ^.siguiente: = nil;
```

```
end;
{**********************************************************}
{**********************************************************}
procedure nacimiento ;
var
  h  : byte;
begin
 if t_reloj  dia_seg then
  begin
     siguiente_llegada(st,llegada,t_reloj);
     nace_evento(nuevo_evento);
     nuevo_evento^.tiempo: = llegada;
     nuevo_evento^.tipo: = 0;
     iserta(nuevo_evento);
  end;
  nace_cliente(cleo);
  llena_cliente(cleo);
  h: = checa_caja(cleo^.tipo);
   mensaje (3,23,'checa caja  = ',8);
   escribe_byte(17,23,h,5,9);
  if h = 0 then forma_cola(cleo)
       else ocupa_caja(h,cleo);
end;{de nacimiento}
{**********************************************************}
{ determina y ejecuta el siguiente evento en la simulacion }
{**********************************************************}
{**********************************************************}
procedure siguiente_evento {(qu :integer)};
var
  evento2 : evento;
  aux  : real;
begin
  mensaje(60,15,'simulacin: ',8);
  escribe_entero(73,15,qu,3,15);
  repeat
   if fin_evento^.siguiente  cab_evento then
    begin
      sig_evento: = fin_evento;
      while sig_evento^.siguiente cab_evento do
       begin
         evento2: =  sig_evento;
         sig_evento: = sig_evento^.siguiente;
```

```
        end;
         t_reloj:=sig_evento^.tiempo;
          aux:=t_reloj;
         Escribe_real(40,20,aux,5,0,7);
         evento2^.siguiente:=cab_evento;
         mensaje (60,19,'tipo de evento',8);
         Escribe_byte(65,20,sig_evento^.tipo,5,7);
         ef:=ef-1;
         escribe_entero(52,19,ef,5,15);
         if sig_evento^.tipo =0 then nacimiento
           else libera_caja(sig_evento^.tipo,cleo);
          muere_evento(sig_evento);
      end;
    until fin_evento^.siguiente = cab_evento;
  end;{siguiente_evento(var nuevo_evento:evento)}
  {*************************************************}
  procedure corrida;
  var
    W_Resultados          : WindowPtr;
    Temp_Atr              : Byte;
    que,qu                : integer;
    tu                    : real;
    desp                  : byte;
  begin
     if banda = false then begin
     assign(ult_par,'ultimo.dat');
     reset(ult_par);
     read(ult_par,parametro);
     close(ult_par);
     end;
        Temp_Atr:=TextAttr;
     TextColor(White); TextBackGround(Blue);
   If not MakeWindow(W_Resultados,1,1,80,25,TRUE,TRUE,FALSE,TextAttr,
TextAttr,TextAttr,'') Then ErrorMem;
    If Not DisplayWindow(W_Resultados) then ErrorMem;
    Presenta_Titulo('CORRIDA DE LA SIMULACION', '',ARRIBA,C_Msg);
              tu:=30;
  mensaje(30,15,' INDIQUE EL NUMERO DE VECES A SIMULAR : ',Text-
Attr);
     desp:=1;
     repeat
        case desp of
```

```
          0 : desp: = 1;
          1 : lee_real_verifica(72,15,tu,3,0,8,desp,1,100);
          2 : desp: = 1;
         end;
     until desp = final;
     Apaga_Mensaje(W_Resultados);
      If not MakeWindow(W_Resultados,1,1,80,25,TRUE,TRUE,FALSE,Text-
Attr,
              TextAttr,TextAttr,'') Then ErrorMem;
    If Not DisplayWindow(W_Resultados) then ErrorMem;
    Presenta_Titulo('CORRIDA DE LA SIMULACION', '',ARRIBA,C_Msg);
     que : = trunc(tu);
    inicia1;
    for qu : = 1 to que do
    begin
     inicia2;
     enuncia_cola;
     siguiente_evento(qu);
     Levanta_est_caja;
    end;
     Apaga_Mensaje(W_Resultados);
     cierra_archivos;
     if banda then begin
      assign(ult_par,'ultimo.dat');
      rewrite(ult_par);
      write(ult_par,parametro);
     close(ult_par);
     end;
  end;
  END.
```

UNIDAD PARA2.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}

Unidad de lectura, modificación y selección de parámetros. La estructura que tiene es la que exige el lenguaje para la construcción de unidades, que se pueden considerar como una extención del mismo.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}

```
UNIT PARA2;
INTERFACE
USES Dos, TPCRT, TPSTRING, TPWINDOW, TPSCREEN, SOUNDS,
    ventanas, GRAPH,typotesi;
{************************************************************}
```

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}

En la declaración de procedimientos, se indican las nuevas instrucciones que estamos conformando y que podran ser invocadas desde cualquier parte del programa.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}

```
{************************************************************}
{          DECLARACION DE PROCEDIMIENTOS               }
{************************************************************}
procedure abre_archivo(var bandera:boolean);
procedure salva_parametro;
Procedure Lee_byte_Verifica(Col,Ren:Byte;
          Var Numero:byte;
          Ancho:Byte;
          Atr:Byte;
          Var Desp:Byte;
          Inf, Sup:Real);
Procedure Lee_Entero_Verifica(Col,Ren:Byte;
          Var Numero:integer;
          Ancho:Byte;
          Atr:Byte;
          Var Desp:Byte;
```

```
            Inf, Sup:Real);
procedure escribe_byte (Col,Ren:Byte;
              Var Numero:byte;
              Ancho:Byte;
              Atr:Byte);
procedure escribe_Entero (Col,Ren:Byte;
              Var Numero:integer;
              Ancho:Byte;
              Atr:Byte);
procedure colores;
procedure presenta_generales ( var parametro:parametros);
Procedure Asigna_Vector_parametro ( var parametro:parametros;
             i:integer;var inf:vector_real);
Procedure Asigna_parametro_Vector(var parametro:parametros;
             i:integer;   var inf  : vector_real);
procedure presenta_operacion ( Xo,Yo:byte; i:integer;
                 var inf     : vector_real;
                 var parametro:parametros);
procedure Lee_operacciones( Xo,Yo : byte;
               var parametro: parametros);
procedure convierte_a_parametro ( var parametro : parametros;
                 var a,b : vector_entero);
procedure convierte_a_vector ( var parametro : parametros;
                 var a,b : vector_entero);
procedure Presenta_tiempo ( Xo,Yo : byte;
               var a,b : vector_entero;
               var parametro : parametros);
procedure Variacion_de_tiempo( Xo,Yo,aux: byte;
                 VAR parametro :parametros);
procedure presenta_clientes(Xo:byte; var parametro:parametros);
procedure Porcentaje_de_clientes(Xo : byte; var parametro : parametros);
procedure presenta_cola(var parametro:parametros);
procedure Atencion_de_colas(var parametro : parametros);
Procedure establece_parametros(var parametro:parametros);
procedure Seleccion_de_disco (var parametro:parametros);
procedure Nuevos_Tiempos;
procedure Nuevos_Tipos_de_cliente;
procedure Nuevos_Tipos_de_operaciones;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
 A continuación se desarrollan los procedimientos enenciados anteriormente.
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

```
{**********************************************************}
IMPLEMENTATION
{**********************************************************}
```

El siguiente procedimiento, abre el archivo de parametros, almacenado en disco y regresa en la variable booleana denominada bandera, como verdadera en caso de haber abierto el archivo, en caso de no existir devuelve la variable como falsa. El siguiente procedimiento almacena en disco en el archivo de parámetros un conjunto de parámetros.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
procedure abre_archivo{(var bandera:boolean)};
begin
  assign(archivo,'parametr.dat');
  {$I-} reset(archivo);{$I+}
  bandera:=Ioresult=0;
end;{abre_archivo}
{**********************************************************}
procedure salva_parametro;
begin
  seek(archivo,filesize(archivo));write(archivo,parametro);
  close(archivo);
end;
{**********************************************************}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Los siguientes dos procedimientos son de lectura, donde se se indica la posicion donde debe leerse (Col y Ren), la variable que se va a leer (número), el ancho del número a leer, el atributo (color) con el que se presentará en pantalla, el nombre de la variable de control de lectura (desp) y los limites del rango en que debe de encontrarse dicha cantidad. Los siguientes dos son de escritura y se solicita desde donde escribir el número, el nombre de la variable númerica a escribir, el color y el ancho del máximo del número.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
Procedure Lee_byte_Verifica{(Col,Ren:Byte;
                Var Numero:byte;
                Ancho:Byte;
                Atr:Byte; Var Desp:Byte;
                Inf, Sup:Real)};
var
aux : real;
begin
   aux:=Numero;
   Lee_Real_Verifica(col,ren,aux,Ancho,0,Atr,Desp,Inf,Sup);
   Numero:= trunc(aux);
end;{Lee_byte_Verifica}
```

```
{**********************************************************}
Procedure Lee_Entero_Verifica{(Col,Ren:Byte;
                Var Numero:integer;
                Ancho:Byte;
                Atr:Byte;
                Var Desp:Byte;
                Inf, Sup:Real)};
var
aux : real;
begin
   aux: = Numero;
   Lee_Real_Verifica(col,ren,aux,Ancho,0,Atr,Desp,Inf,Sup);
   Numero: = trunc(aux);
end;{Lee_Entero_Verifica}
{**********************************************************}
procedure escribe_byte {(Col,Ren:Byte;
                Var Numero:byte;
                Ancho:Byte;
                Atr:Byte)};
var
 aux : real;
begin
   aux: = Numero;
   Escribe_real (Col,Ren,aux,Ancho,0,Atr);
end; {Escribe_byte}
{**********************************************************}
procedure escribe_Entero {(Col,Ren:Byte;
                Var Numero:integer;
                Ancho:Byte;
                Atr:Byte)};
var
 aux : real;
begin
   aux: = Numero;
   Escribe_real (Col,Ren,aux,Ancho,0,Atr);
end; {Escribe_byte}
{**********************************************************}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
Asigna colores a variables que estaremos utilizando en los demas procedimientos.
{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
procedure colores;
begin
```

```
   C_Msg: = PC(YELLOW,RED); C_Num: = PC(WHite,Blue);
   C_Lee: = PC(White,Green); C_Tit: = PC(Red,LightGray);
end;{de colores}
{**********************************************************}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Este procedimiento escribe en un recuadro los valores generales de un parámetro, que puede ser utilizado para presentar los valores actuales de un conjunto de parámetros o bien para solicitar la modificación de los mismos.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
procedure presenta_generales {( var parametro:parametros)};
begin
 with parametro do
  begin
   Mensaje(3,6,'No. D E  C A J A S        :',C_Msg);
   Mensaje(3,7,'No. DE CLIENTES DIFERENTES   :',C_Msg);
   Mensaje(3,8,'No. DE OPERACIONES DIFERENTES :',C_Msg);
   Mensaje(3,9,'No. DE VARIACIONES DE LLEGADAS:',C_Msg);
   Escribe_byte(35,6,nume_de_cajas,2,C_Lee);
   Escribe_byte(35,7,nume_tipos_clientes,2,C_Lee);
   Escribe_byte(35,8,nume_de_operaciones,2,C_Lee);
   Escribe_byte(35,9,nume_de_cambios_de_llegada,2,C_Lee);
  end;
end;{presenta_generales}
{**********************************************************}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Los siguientes dos procedimientos, son asignaciones de un vector de reales, a los parámetros referidos a las operaciones a realizar por los clientes. Su unica finalidad es simplificar la notación de captura de estos datos en los procedimientos posteriores.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
Procedure Asigna_Vector_parametro {( var parametro:parametros;
                i:integer;var inf: vector_real)};
begin
  with parametro do
  begin
   with operacion[i] do
    begin
     case  nume_tipos_clientes of
```

```
        1 : begin
                    inf[1]:=t_m_s[1]; inf[2]:=m_t_m[1]; inf[3]:=dis_t_m_s[1];
inf[4]:=valor_medio[1]; inf[5]:=dis_v_m[1]; inf[6]:=porcentaje[1];
            end;
        2 : begin
                    inf[1]:=t_m_s[1]; inf[3]:=m_t_m[1]; inf[5]:=dis_t_m_s[1];
inf[7]:=valor_medio[1]; inf[9]:=dis_v_m[1]; inf[11]:=porcentaje[1];
inf[2]:=t_m_s[2];      inf[4]:=m_t_m[2];      inf[6]:=dis_t_m_s[2];
inf[8]:=valor_medio[2]; inf[10]:=dis_v_m[2]; inf[12]:=porcentaje[2];
            end;
        3 : begin
                    inf[1]:=t_m_s[1]; inf[4]:=m_t_m[1]; inf[7]:=dis_t_m_s[1];
inf[10]:=valor_medio[1]; inf[13]:=dis_v_m[1]; inf[16]:=porcentaje[1];
inf[2]:=t_m_s[2];      inf[5]:=m_t_m[2];      inf[8]:=dis_t_m_s[2];
inf[11]:=valor_medio[2]; inf[14]:=dis_v_m[2]; inf[17]:=porcentaje[2];
inf[3]:=t_m_s[3];      inf[6]:=m_t_m[3];      inf[9]:=dis_t_m_s[3];
inf[12]:=valor_medio[3]; inf[15]:=dis_v_m[3]; inf[18]:=porcentaje[3];
            end;
       end; {del case}
      end;{operacion}
    end;{parametro}
  end; {de asigna_vector_parametro}
  {*********************************************************}
  Procedure Asigna_parametro_Vector{(var parametro:parametros;
                i:integer;   var inf  : vector_real)};
  begin
    with parametro do
    begin
     with operacion[i] do
     begin
      case  nume_tipos_clientes of
       1 : begin
                    t_m_s[1]:=inf[1]; m_t_m[1]:=inf[2]; dis_t_m_s[1]:=inf[3];
valor_medio[1]:=inf[4]; dis_v_m[1]:=inf[5]; porcentaje[1]:=inf[6];
            end;
       2 : begin
                    t_m_s[1]:=inf[1]; m_t_m[1]:=inf[3]; dis_t_m_s[1]:=inf[5];
valor_medio[1]:=inf[7]; dis_v_m[1]:=inf[9]; porcentaje[1]:=inf[11];
t_m_s[2]:=inf[2];      m_t_m[2]:=inf[4];      dis_t_m_s[2]:=inf[6];
valor_medio[2]:=inf[8]; dis_v_m[2]:=inf[10]; porcentaje[2]:=inf[12];
            end;
       3 : begin
                    t_m_s[1]:=inf[1]; m_t_m[1]:=inf[4]; dis_t_m_s[1]:=inf[7];
```

```
valor_medio[1]:=inf[10]; dis_v_m[1]:=inf[13]; porcentaje[1]:=inf[16];
t_m_s[2]:=inf[2];       m_t_m[2]:=inf[5];       dis_t_m_s[2]:=inf[8];
valor_medio[2]:=inf[11]; dis_v_m[2]:=inf[14]; porcentaje[2]:=inf[17];
t_m_s[3]:=inf[3];       m_t_m[3]:=inf[6];       dis_t_m_s[3]:=inf[9];
valor_medio[3]:=inf[12]; dis_v_m[3]:=inf[15]; porcentaje[3]:=inf[18];
          end;
      end; {del case}
     end;{operacion}
   end;{parametro}
 end; {de asigna_parametro_vector}
 {************************************************************}
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Este procedimiento, presenta en pantalla un recuadro donde solicita al usuario, los parámetros referentes a las operaciones, por cada tipo de clientes (tiempos medios de servicio, etc.), el siguiente recaba esa información.

```
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
 procedure presenta_operacion {( Xo,Yo:byte; i:integer;
                    var inf     : vector_real;
                    var parametro:parametros)};
 var ind,cont  : byte;
 begin
  with parametro do
   begin
    with operacion[i] do
     begin
      mensaje (Xo+1,Yo+1,'    N O M B R E',C_Msg);
      mensaje (Xo+20,Yo+1,nombre,C_Lee);
      if nume_tipos_clientes then
      mensaje (Xo+22,Yo+3,' C L I E N T E S ',C_Lee);
      case nume_tipos_clientes of
        1 : mensaje (Xo+25,Yo+4,' U N I C O',C_Msg);
        2 : begin
            mensaje (Xo+22,Yo+4,'TIPO I',C_Msg);
            mensaje (Xo+33,Yo+4,'TIPO II',C_Msg);
            T_Vertical(Xo+31,Yo+5,23,C_Tit);
           end;
        3 : begin
            mensaje (Xo+27,Yo+3,' C L I E N T E S ',C_Lee);
            mensaje (Xo+22,Yo+4,'TIPO I',C_Msg);
            mensaje (Xo+32,Yo+4,'TIPO II',C_Msg);
            mensaje (Xo+42,Yo+4,'TIPO III',C_Msg);
            T_Vertical(Xo+30,Yo+5,23,C_Tit);
```

```
        T_Vertical(Xo + 40,Yo + 5,23,C_Tit);
      end;
   end;{del case}
   mensaje (Xo + 1,Yo + 6,'TIEMPO MED. DE SERV.',C_Msg);
   mensaje (Xo + 1,Yo + 7,'DESV. DE TIEMPO MED.',C_Msg);
   mensaje (Xo + 1,Yo + 8,'DISTRIBUCION      ',C_Msg);
   mensaje (Xo + 1,Yo + 9,'VALOR MEDIO DE OPER.',C_Msg);
   mensaje (Xo + 1,Yo + 10,'DESV. DE VALOR MEDIO',C_Msg);
   mensaje (Xo + 1,Yo + 11,'PORCENTAJE       ',C_Msg);
      case nume_tipos_clientes of
       1 : for ind: = 1 to 6 do
             Escribe_Real(Xo + 27,Yo + 5 + ind,inf[ind],6,1,C_Num);
       2 : begin
           for ind: = 1 to 12 do
            begin
            if ind mod 2  = 1then cont: =  22 else cont: = 33;
           Escribe_real(Xo + cont,Yo + 6 + (ind-1) div 2 ,inf[ind],6,1,C_Num );
          end;
          end;
       3 : begin
         for ind: = 1 to 18 do
         begin
          if ind mod 3  =  1 then cont: = 22 else
          if ind mod 3  =  2 then cont: = 32 else cont: = 42;
          Escribe_real(Xo + cont,Yo + 6 + (ind-1) div 3 ,inf[ind],6,1,C_Num);
         end;
         end;
      end;{ del case}
  end; {del with de operacion}
end; {del with de parametro}
end; {presenta_operacion}
{*************************************************************}
procedure Lee_operacciones {( Xo,Yo : byte; var parametro: parametros)};
 var
  W_oper   : WindowPtr;
  nombre   : string;
  cont,desp                : Byte;
  ind,cont1                : integer;
  inf                      : vector_real;
  cot                      : real;
begin
 with parametro do
```

```
begin
 for cont1 := 1 to nume_de_operaciones do begin
   Asigna_Vector_parametro( parametro,cont1,inf);
  case nume_tipos_clientes of
   1 : desp:=41;
   2 : desp:=40;
   3 : desp:=53;
  end;
                                        If not MakeWindow(W_oper,Xo,Yo,Xo + desp,23,TRUE,TRUE,FALSE,TextAttr, TextAttr,TextAttr,'DEFINICION DE OPERACIONES') Then ErrorMem;
   If Not DisplayWindow(W_oper) then ErrorMem;
        T_horizontal(Xo + 1,Xo + desp-1,Yo + 5,C_Tit);
        T_horizontal(Xo + 1,Xo + desp-1,Yo + 2,C_Tit);
  presenta_operacion(Xo,Yo,cont1,inf,parametro);
   desp:= 1;
   nombre:= operacion[cont1].nombre;
   repeat
    case desp of
    0: desp:= 1;
    1: Lee_cadena( Xo + 18,Yo + 1,nombre,20,C_Lee,desp);
    2: desp:= 1;
    end;{del case}
   until desp = final;
   operacion[cont1].nombre:= copy(nombre,1,length(nombre));
   desp:= 1;
      case nume_tipos_clientes of
       1 : begin
          repeat
           if desp = 7 then desp:= 1;
           if desp = 0 then desp:= 6;
           if desp = 6 then cot:= 100 else cot:= 1600;
Lee_Real_Verifica(Xo + 27,Yo + 5 + desp,inf[desp],6,1,C_Num,desp,0,cot)
          until desp = final;
         end;
       2 : begin
          repeat
           if desp = 0 then desp:= 12;
           if desp = 13 then desp:= 1;
           if desp mod 2 = 1 then cont:= 22 else cont:= 33;
           ind:= (desp-1) div 2;
           if ind = 5 then cot:= 100 else cot := 1600;
```

```
Lee_Real_Verifica(Xo + cont,Yo + 6 + ind,inf[desp],6,1,C_Num, desp,0,cot)
        until desp = final;
          end;
        3 : begin
        repeat
          if desp = 0 then desp: = 18;
          if desp = 19 then desp: = 1;
          if desp mod 3 = 1 then cont: = 22 else
          if desp mod 3 = 2 then cont: = 32 else cont: = 42;
          ind : = (desp-1)div 3;
          if ind = 5 then cot: = 100 else cot: = 1600;
    Lee_Real_Verifica(Xo + cont,Yo + 6 + ind,inf[desp],6,1,C_Num, desp,0,cot);
        until desp = final;
          end;
      end;{ del case}
    Apaga_mensaje(W_oper);
    Asigna_parametro_Vector( parametro,cont1,inf);
   end;{del for}
  end; {del with}
  end;{Lee_operaciones}
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Este procedimiento combierte un vector de enteros, con los cuales vamos a recoger las horas de cambio de las variaciones de los tiempos medios de llegada, (en horas y minutos) a un solo valor que correspoonde a un equivalente en segundos, que es la unidad de medición en el sistema. Mientras que el siguiente realiza la converción a la inversa.

```
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  procedure convierte_a_parametro {( var parametro : parametros;
                                     var a,b : vector_entero)};
  var i,j  : integer;
  begin
   with parametro do
    begin
     for i: = 1 to nume_de_cambios_de_llegada do
       t_de_llegadas[i].hasta: = (a[i]-8)*3600 + b[i]*60;
    end; {del with de parametro}
  end;{de convierte_a_parametro}
  {************************************************************}
  procedure convierte_a_vector {( var parametro : parametros;
                    var a,b : vector_entero)};
  var i,j  : integer;
  begin
```

```
a[0]:=8; b[0]:=0;
with parametro do
 begin
  for i:=1 to nume_de_cambios_de_llegada do
    begin
      a[i]:= 8 + t_de_llegadas[i].hasta div 3600;
      b[i]:= t_de_llegadas[i].hasta mod 3600;
      b[i]:= b[i] div 60;
    end; {del for}
 end; {del with de parametro}
end;{de convierte_a_vector}
{**********************************************************}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Este procedimiento utiliza uno de los anteriores, para transformar un vector de segundos en su equivalente de horas y minutos, presenta un recuadro para solicitar modificaciones al actual conjunto de parámetro relacionados con los tiempos de llegada, el siguiente procedimiento realiza la captura de modificaciones, cuidando que no se presenten traslapes, despues de la captura, transforma los tiempos capturados a segundos.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
procedure Presenta_tiempo {( Xo,Yo : byte;
                    var a,b : vector_entero;
                    var parametro : parametros)};
var
cont    : byte;
begin
  mensaje(Xo+2,Yo,'D E L A S   A  L A S   T. MEDIO',C_Msg);
  Yo:=Yo+1;
  convierte_a_vector(parametro,a,b);
  with parametro do
  begin
   for cont:=1 to nume_de_cambios_de_llegada do begin
     mensaje(Xo+3,Yo+cont,'Hrs  Min   Hrs  Min      p/min',C_Msg);
     Escribe_Entero(Xo+14,Yo+cont,a[cont],2,C_Num);
     Escribe_Entero(Xo+20,Yo+cont,b[cont],2,C_Num);
     Escribe_Real(Xo+30,Yo+cont, t_de_llegadas[cont].c_p_m ,5,2,C_Num);
   end;
 a[nume_de_cambios_de_llegada]:=14; b[nume_de_cambios_de_llegada]:=0;
   T_Vertical(Xo+12,Yo,Yo+nume_de_cambios_de_llegada,C_Tit);
   T_Vertical(Xo+28,Yo,Yo+nume_de_cambios_de_llegada,C_Tit);
   Escribe_Entero(Xo+1,YO+1,a[0],2,C_Num);
   Escribe_Entero(Xo+7,YO+1,b[0],2,C_Num);
```

```
  end;
 end; {de Presenta_tiempo}
 {*************************************************************}
 procedure Variacion_de_tiempo {( Xo,Yo,aux: byte;
                  VAR parametro :parametros)};
  var
  cont2, cont,desp              : Byte;
   ind                    : integer;
   inf                    : real;
   a,b                    : vector_entero;
   begin
    parametro.nume_de_cambios_de_llegada: = aux;
    presenta_tiempo(Xo,Yo,a,b,parametro);
   with parametro do
   begin
    Yo: = Yo + 1;
    cont: = 1; desp: = 1;
    repeat
      if desp = 0 then desp: = 1;
      ind : = desp mod 3;
      case ind of
       1 : begin
           if b[cont-1] = 59 then inf: = a[cont-1] + 1{cota inferior}
             else inf: = a[cont-1];
          if cont < nume_de_cambios_de_llegada then begin
          cont2: = desp;
                Lee_Entero_verifica(Xo + 14,Yo + cont,a[cont],2,C_Num,
desp,inf,a[cont + 1]);
          Escribe_Entero(Xo + 1,Yo + cont + 1,a[cont],2,C_Num);
          if cont2 > desp then
                  if cont  = 1 then begin  cont: = nume_de_cambios_de_llegada;
desp: = nume_de_cambios_de_llegada*3 end
                else cont: = cont-1;
          end else begin
          a[cont]: = 14; Escribe_Entero(Xo + 14,YO + cont,a[cont],2,C_Num);
          desp: = desp + 1;
          end;
         end;
       2 : begin
          if a[cont] > a[cont-1] then inf: = 0
           else inf : = b[cont-1];
         if cont < nume_de_cambios_de_llegada then begin
```

```
          Lee_Entero_verifica(Xo+20,Yo+cont,b[cont],2,C_Num,desp,inf,59);
          Escribe_Entero(Xo+7,YO+cont+1,b[cont],2,C_Num);
          end else begin
          b[cont]:=0; Escribe_Entero(Xo+20,Yo+cont,b[cont],2,C_Num);
          desp:=desp+1
          end;
         end;
       0 : begin
           cont2:= desp;
  Lee_Real_Verifica( Xo+30,Yo+cont, t_de_llegadas[cont].c_p_m
,5,2,C_Num,desp,0,20);
              if (cont2<desp) and (desp <final) then if cont < nume_de_cam-
bios_de_llegada then cont:=cont+1 elsebegin desp:=1; cont:=1;end;
           if (cont2>desp) and (nume_de_cambios_de_llegada=cont) and
            (nume_de_cambios_de_llegada>1) then cont:=cont-1;
          end;
      end; {del case}
     until desp=final;
     convierte_a_parametro ( parametro,a,b );
    end;{del with}
  end;{Variacion_de_tiempo}
  {**********************************************************}
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  Este procedimiento presenta un recuadro en pantalla, para solicitar al usuario
que indique la proporción en que se presentan los distintos tipos de clientes. El
siguiente procedimiento captura esa información, garantizando que la todos los
tipos de clientes tengan un valor positivo de proporción y que la suma de todos de
100.
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  procedure presenta_clientes {(Xo:byte;var parametro:parametros)};
  var
   cont,desp   : byte;
  begin
   mensaje(Xo,5,'CLIENTE   %',C_Msg);
    T_Horizontal(Xo,Xo+14,6,C_Tit);
   with parametro do
   begin
    cont:=7; desp:=por_cliente[1];
   mensaje(Xo,cont,'TIPO I',C_Num);
   Escribe_byte(Xo+10,cont,por_cliente[1],2,C_Num);
   cont:=cont+1;
   mensaje(Xo,cont,'TIPO II',C_Num);
```

```
   if nume_tipos_clientes =3 then begin
     Escribe_byte(Xo + 10,cont,por_cliente[2],2,C_Num);
     cont: = cont + 1;  desp: = desp + por_cliente[2];
   mensaje(Xo,cont,'TIPO III',C_Num);
   end;
   desp: = 100-desp;
   Escribe_byte(Xo + 10,cont,desp,2,C_Num);
  end; {del with}
end;{de presenta_clientes}
{**********************************************************}
procedure Porcentaje_de_clientes{(Xo: byte; var parametro :  parametros)};
var
 cadena: string[11];
 cont2,cont1,cont,desp  : Byte;
 inf              : real;
 begin
  presenta_Clientes(65,parametro);
  with parametro do
  begin
   desp: = 1; cont: = 7;
   repeat
     if desp in [nume_tipos_clientes,0] then begin
       desp: = 1; cont: = 7 end
       else begin
       case nume_tipos_clientes of
         2 : cont1: = 100;
         3 : if desp = 1 then cont1: = 100-por_cliente[2]
              else cont1: = 100-por_cliente[1];
        end;{del case}
       cont2: = desp;
      lee_byte_Verifica(Xo + 10,cont,por_cliente[desp],2,C_Lee, desp,1,cont1);
      if desp < final then begin
       cont1: = cont1-por_cliente[desp-1];
       if cont2 < desp then  cont: = cont + 1
       else  cont: = cont-1;
 Escribe_byte(Xo + 10,6 + nume_tipos_clientes,cont1,2,C_Num);
      end;
     end;
   until desp = final;
  end;{del with}
end;{Porcentaje_de_clientes}
```

```
{**********************************************************}
{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```
Este procedimiento presenta en pantalla un recuadro que será utilizado en el siguiente procedimiento, donde se solicita al usuario que indique de que caja a que caja se dará servicio a los diversos clientes, el procedimiento de captura asegura que todos los distintos tipos de clientes tengan almenos una de las cajas para ser atendidos.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
procedure presenta_cola {(var parametro:parametros)};
var
 cont,desp   : byte;
begin
  mensaje(41,5,'DE LA CAJA  A LA CAJA',C_Msg);
  T_Horizontal(41,61,6,C_Tit);
  with parametro do
  begin
   cont: = 7;
  mensaje(46,cont,' 1',C_Num);
  Escribe_byte(57,cont,cambio_de_caja[1],2,C_Num);
  cont: = cont + 1; desp: = cambio_de_caja[1] + 1;
  Escribe_byte(46,cont,desp,2,C_Num);
  if nume_tipos_clientes  = 3 then begin
    Escribe_byte(57,cont,cambio_de_caja[2],2,C_Num);
    cont: = cont + 1; desp: = cambio_de_caja[2] + 1;
    Escribe_byte(46,cont,desp,2,C_Num);
  end;
  Escribe_byte(57,cont,nume_de_cajas,2,C_Num);
 end; {del with}
end;{de presenta_cola}
{**********************************************************}
procedure Atencion_de_colas {(var parametro : parametros)};
var
 cadena: string[11];
 cont2,cont1,cont,desp  : Byte;
 inf                    : real;
 begin
  presenta_cola(parametro);
  with parametro do
  begin
   desp: = 1; cont: = 7;
   repeat
     if desp in [nume_tipos_clientes,0] then begin
```

```
        desp:=1; cont:=7 end
        else begin
       if desp=1 then inf:=1 else inf:=cambio_de_caja[desp-1]+1;
        if nume_tipos_clientes=2 then cont1:=nume_de_cajas-1 else if desp=1
then cont1:=cambio_de_caja[2]-1          else cont1:=nume_de_cajas-1;
        cont2:=desp;
      lee_byte_Verifica(57,cont,cambio_de_caja[desp],2,C_Lee, desp,inf,cont1);
       if desp<final then begin if cont2<desp  then begin
          cont:=cont+1; cont1:=cambio_de_caja[desp-1]+1;
          Escribe_byte(46,cont,cont1,2,C_Num);
        end else cont:=cont-1;
       end;
      end;
    until desp=final;
   end;{del with}
  end;{Atencion_de_colas}
  {**********************************************************}
  {{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  El siguiente procedimiento establece nuevos parámetros.
  {{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  Procedure establece_parametros {(var parametro:parametros)};
  var
    W_Resultados, W_Res,W_clie,
    W_Colas, W_Oper, W_time   : WindowPtr;
    Temp_Atr              : Byte;
    Desp ,cont,derecho      : Byte;
     nombres              : string;
    Resp, Tipo_Datos        : Char;
    aux                : Real;
    cambio               : camb_caja;
    llegadas             : vector_tiempo_medio_de_llegadas;
    operacion             : vector_de_operaciones;
  begin
    banda:=true;
    Temp_Atr:=TextAttr;
    TextColor(White); TextBackGround(Blue);
      If not MakeWindow(W_Resultados,1,1,80,25,TRUE,TRUE,FALSE,Text-
Attr,TextAttr,TextAttr,'') Then ErrorMem;
    If Not DisplayWindow(W_Resultados) then ErrorMem;
  Presenta_Titulo('MODIFICACION DE PARAMETROS','',AR-
RIBA,C_Msg);
    If not MakeWindow(W_Res,2,5,38,10,TRUE,TRUE,FALSE,TextAttr,
```

```
            TextAttr,TextAttr,'GENERALES') Then ErrorMem;
    If Not DisplayWindow(W_Res) then ErrorMem;
    presenta_generales(parametro);
    desp:= 1;
    with parametro do
    begin
     repeat
       case desp of
        0: desp:= 1;
      1 : lee_byte_Verifica(35,6,nume_de_cajas,2,C_Lee, desp,1,nume_de_cajas);
      2 : begin
            if nume_de_cajasnume_tipos_clientes then derecho:= num_de_clientes
else derecho:= nume_de_cajas;
   lee_byte_Verifica(35,7,nume_tipos_clientes,2,C_Lee, desp,1,derecho);
          end;
        3  :  lee_byte_Verifica(35,8,nume_de_operaciones,2,C_Lee,
desp,1,num_de_oper);
     4 : lee_byte_Verifica(35,9,nume_de_cambios_de_llegada,2,C_Lee,
desp,1,nitll);
        5 : desp:= 1
        end;
     until desp = final;
    if nume_tipos_clientes  1 then begin
     cont:= 7 + nume_tipos_clientes;
    If not MakeWindow(W_Colas,40,5,62,cont,TRUE,TRUE,FALSE,TextAttr,
              TextAttr,TextAttr,'ATENCION EN CAJAS') Then ErrorMem;
    If Not DisplayWindow(W_Colas) then ErrorMem;
     atencion_de_colas( parametro);
    If not MakeWindow(W_Clie,64,5,79,cont,TRUE,TRUE,FALSE,TextAttr,
              TextAttr,TextAttr,'  PORCENTAJES  ') Then ErrorMem;
    If Not DisplayWindow(W_Clie) then ErrorMem;
    presenta_clientes(65,parametro);
    Porcentaje_de_clientes(65,parametro);
    end;
          If not    MakeWindow(W_Time,3,11,50,14 + nume_de_cam-
bios_de_llegada,TRUE,TRUE,FALSE,TextAttr,TextAttr,TextAttr,'TIEMPOS
MEDIOS DE SERVICIO') Then ErrorMem;
    If Not DisplayWindow(W_Time) then ErrorMem;
    Variacion_de_tiempo(3,12, nume_de_cambios_de_llegada,parametro);
    Lee_operacciones( 15,10,parametro);
    Apaga_Mensaje(W_time);
    if nume_tipos_clientes  1 then begin Apaga_Mensaje(W_Colas);
```

```
                Apaga_Mensaje(W_Clie) end;
    Apaga_Mensaje(W_Res);
    salva_parametro;
    Apaga_Mensaje(W_Resultados);
  end; {del with}
  end;{de establece parametros}
  {**********************************************************}
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
```

Este procedimiento, lee de disco un conjunto de parametros, los presenta en pantalla y pregunta al usuario, si se queda con esos parametros o selecciona otro de disco.

```
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
  procedure Seleccion_de_disco {(var parametro:parametros)};
  var
    W_Resultados, W_Res,W_clie,W_maco,
    W_Colas, W_Oper, W_time   : WindowPtr;
    Temp_Atr              : Byte;
    Desp                : Byte;
    cont,derecho          : Byte;
    Resp, Tipo_Datos        : Char;
    aux                 : Real;
    cambio                : camb_caja;
    llegadas              : vector_tiempo_medio_de_llegadas;
    operacion             : vector_de_operaciones;
    bandera               : boolean;
  begin
    banda: = true;
    Temp_Atr: = TextAttr;
    TextColor(White); TextBackGround(Blue);
    If not    MakeWindow(W_Resultados,1,1,80,25,TRUE,TRUE,FALSE,Text-
Attr, TextAttr,TextAttr,'') Then ErrorMem;
    If Not DisplayWindow(W_Resultados) then ErrorMem;
     Presenta_Titulo('SELECCION DE PARAMETROS DE ARCHIVO EN
DISCO', '',ARRIBA,C_Msg);
    abre_archivo(bandera);
    resp: = 'n';
   while resp in ['n','N'] do
    begin
    if not eof(archivo) then begin
     read(archivo,parametro);
       If not    MakeWindow(W_Res,2,5,38,10,TRUE,TRUE,FALSE,TextAttr,
TextAttr,TextAttr,'GENERALES') Then ErrorMem;
```

```
If Not DisplayWindow(W_Res) then ErrorMem;
presenta_generales(parametro);
if parametro.nume_tipos_clientes 1 then begin
cont: = 7 + parametro.nume_tipos_clientes;
 If not MakeWindow (W_Colas,40,5,62,cont,TRUE,TRUE,FALSE ,Text-
Attr, TextAttr,TextAttr,'ATENCION EN CAJAS') Then ErrorMem;
If Not DisplayWindow(W_Colas) then ErrorMem;
presenta_cola( parametro);
 If not MakeWindow(W_Clie,64,5,79,cont,TRUE,TRUE,FALSE,Text-
Attr, TextAttr,TextAttr,' PORCENTAJES ') Then ErrorMem;
If Not DisplayWindow(W_Clie) then ErrorMem;
presenta_clientes(65,parametro);
end;
 If not MakeWindow(W_Time,3,11,50,14 + parametro.nume_de_cam-
bios_de_ llegada,TRUE,TRUE,FALSE,TextAttr, TextAttr, TextAttr, 'TIEMPOS
MEDIOS DE SERVICIO') Then ErrorMem;
If Not DisplayWindow(W_Time) then ErrorMem;
 Variacion_de_tiempo(3,12,parametro.nume_de_ cambios_de_llegada,
parametro);
Lee_operacciones( 15,10,parametro);
 If not MakeWindow(W_Maco,1,20,80,25,TRUE,TRUE,FALSE,Text-
Attr, TextAttr,TextAttr,'') Then ErrorMem;
If Not DisplayWindow(W_Maco) then ErrorMem;
 Presenta_Titulo('S PARA ACEPTAR ESTE PARAMETRO N PARA
CONTINUAR', '',ABAJO,C_Msg);
repeat
 resp: = readkey;
until resp in ['S','s','n','N'];
Apaga_mensaje(W_maco);
Apaga_Mensaje(W_time);
if parametro.nume_tipos_clientes 1 then begin Apaga_Mensaje(W_Colas);
Apaga_Mensaje(W_Clie) end; Apaga_Mensaje(W_Res);
end{delthen} else begin
If not MakeWindow(W_Maco,1,20,80,25,TRUE,TRUE,FALSE,TextAttr,
TextAttr,TextAttr,'') Then ErrorMem;
If Not DisplayWindow(W_Maco) then ErrorMem;
Presenta_Titulo('FIN DE ARCHIVO, "N" PARA REVISAR DE NUEVO "S"
PARA SALIR', '',ABAJO,C_Msg);
repeat
 resp: = readkey;
until resp in ['S','s','n','N'] ;
if resp in ['N','n'] then seek(archivo,0);
```

```
    Apaga_Mensaje(W_maco);
    end; {del else}
  end; {del while}
   Apaga_mensaje(W_Resultados);
 end;{de Seleccion_de_disco}
 {**********************************************************}
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Este procedimiento, modifica del actual conjunto de parámetros, los relativos al número de variaciones de tiempos de llegada de los clientes y desde luego los tiempos medios de llegada.

```
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
 procedure Nuevos_Tiempos;
 var
   W_Oper, W_time   : WindowPtr;
   Temp_Atr         : Byte;
   Desp             : Byte;
   cont,derecho     : Byte;
   nombres          : string;
   Resp, Tipo_Datos : Char;
   aux              : byte;
   cambio           : camb_caja;
   llegadas         : vector_tiempo_medio_de_llegadas;
   operacion        : vector_de_operaciones;
 begin
   banda:=true;
   Temp_Atr:=TextAttr;
   TextColor(White); TextBackGround(Blue);
    If not MakeWindow(W_Oper,1,1,80,25,TRUE,TRUE,FALSE,TextAttr,
          TextAttr,TextAttr,'') Then ErrorMem;
   If Not DisplayWindow(W_Oper) then ErrorMem;
    Presenta_Titulo('MODIFICACION DE VARIACION DE TIEMPOS DE
LLEGADAS','',ARRIBA,C_Msg);
      mensaje(4,7,'INDIQUE EL NUMERO DE VARIACIONES DE TIEM-
POS DE LLEGADAS:',C_Tit);
   desp:=1;
   escribe_byte(62,7,parametro.nume_de_cambios_de_llegada, 2,C_Num);
   repeat
    case desp of
      0: desp:=1;
       1   :     Lee_Byte_Verifica(62,7,parametro.nume_de_cam-
bios_de_llegada,2,C_Num, desp,1,nitll);
      2: desp:=1;
```

```
    end;
  until desp = final;
    If not MakeWindow(W_Time,3,11,50,14 + parametro.nume_de_cambios_de_llegada,TRUE,TRUE,FALSE,TextAttr, TextAttr,TextAttr, 'TIEMPOS MEDIOS DE SERVICIO') Then ErrorMem;
  If Not DisplayWindow(W_Time) then ErrorMem;
      Variacion_de_tiempo(3,12,parametro.nume_de_cambios_de_llegada,parametro);
  Apaga_mensaje(W_Time);
  salva_parametro;
  Apaga_Mensaje(W_Oper);
end;{Nuevos_tiempos}
{**********************************************************}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
Este procedimiento cambia exclusivamente, el número de clientes diferentes y la atención en cajas de la nueva definición de clientes.
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
procedure Nuevos_Tipos_de_cliente;
var
  W_Oper, W_Colas,W_clie   : WindowPtr;
  Temp_Atr          : Byte;
  Desp              : Byte;
  cont,derecho,aux1    : Byte;
  nombres           : string;
  Resp, Tipo_Datos      : Char;
  cambio            : camb_caja;
  llegadas          : vector_tiempo_medio_de_llegadas;
  operacion         : vector_de_operaciones;
begin
  banda: = true;
  Temp_Atr: = TextAttr;
  TextColor(White); TextBackGround(Blue);
    If not MakeWindow(W_Oper,1,1,80,25,TRUE,TRUE,FALSE, TextAttr, TextAttr,TextAttr,'') Then ErrorMem;
  If Not DisplayWindow(W_Oper) then ErrorMem;
    Presenta_Titulo('MODIFICACION DE DISTINTOS TIPOS DE CLIENTES', '',ARRIBA,C_Msg);
    mensaje(4,7,'INDIQUE EL NUMERO DE DIFERENTES TIPOS DE CLIENTES',C_Tit);
  desp: = 1;
  escribe_byte(61,7,parametro.nume_tipos_clientes,2,C_Num);
  repeat
```

```
      case desp of
        0 : desp:= 1;
        1 : Lee_Byte_Verifica (62,7,aux1,1,C_Num,desp,1,3);
        2 : desp:= 1;
       end;
     until desp = final;
     parametro.nume_tipos_clientes:= aux1;
     if aux1  1 then begin
      cont:= 7 + aux1;
     If not   MakeWindow(W_Colas,40,5,62,cont,TRUE,TRUE,FALSE,TextAttr,
TextAttr,TextAttr,'ATENCIÓN EN CAJAS') Then ErrorMem;
     If Not DisplayWindow(W_Colas) then ErrorMem;
      atencion_de_colas(parametro);
      cont:= 7 + aux1;
      If not    MakeWindow(W_Clie,64,5,79,cont,TRUE,TRUE,FALSE,TextAttr,
TextAttr,TextAttr,'  PORCENTAJES  ') Then ErrorMem;
     If Not DisplayWindow(W_Clie) then ErrorMem;
     presenta_clientes(65,parametro);
     Porcentaje_de_clientes(65,parametro);
      Apaga_mensaje(W_Clie);
      Apaga_Mensaje(W_colas);
     end;
       salva_parametro;
    Apaga_Mensaje(W_Oper);
  end;{ Nuevos_Tipos_de_cliente}
  {**********************************************************}
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  Este procedimiento modifica exclucivamente, el número de operaciones y sus
especificaciones del pararmetro actual.
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  procedure Nuevos_Tipos_de_operaciones;
  var
    W_Oper, W_Colas,W_clie    : WindowPtr;
    Temp_Atr              : Byte;
    Desp                : Byte;
    cont,derecho            : Byte;
    nombres               : string;
    Resp, Tipo_Datos        : Char;
    aux1                : byte;
    cambio                : camb_caja;
    llegadas              : vector_tiempo_medio_de_llegadas;
    operacion             : vector_de_operaciones;
```

```
begin
   banda: = true;
   Temp_Atr: = TextAttr;
   TextColor(White); TextBackGround(Blue);
    If not MakeWindow(W_Oper,1,1,80,25,TRUE,TRUE,FALSE, TextAttr,'')
Then ErrorMem;
   If Not DisplayWindow(W_Oper) then ErrorMem;
    Presenta_Titulo('MODIFICACION DE OPERACIONES', '',AR-
RIBA,C_Msg);
        mensaje (4,7,'INDIQUE EL NUMERO TOTAL DE
OPERACIONES',C_Tit);
   desp: = 1; escribe_byte(43,7,parametro.nume_de_operaciones,2,C_Num);
   repeat
    case desp of
      0 : desp: = 1;
      1 : Lee_Byte_Verifica (43,7,aux1,2,C_Num,desp,1,10);
      2 : desp: = 1;
     end;
   until desp = final;
   parametro.nume_de_operaciones: = aux1;
    Lee_operacciones( 15,10,parametro);
  Apaga_Mensaje(W_Oper);
   salva_parametro;
 end;{ Nuevos_Tipos_de_operaciones }
 {**********************************************************}
 {**********************************************************}
 END.
```

UNIDAD ESTADIS.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
Esta unidad tiene procedimientos para trabajar en modo gráfico y los relacionados con procesar los archivos formados en la simulación.
{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}

```
UNIT ESTADIS;
INTERFACE
USES Dos, TPCRT, TPSTRING, TPWINDOW, TPSCREEN, SOUNDS,para2,
   cuatro, ventanas, GRAPH,typotesi;
{**************************************************************}
{**************************************************************}
{          DECLARACION DE PROCEDIMIENTOS          }
{**************************************************************}
Procedure Inicia_G;
Procedure Termina_G;
procedure maximo (var x:frecuen;i:byte;var ym:integer);
procedure barras (var x : frecuen; i : byte;s1,s2:string);
procedure abre_arch_clientes;
procedure abre_arch_oper;
procedure abre_Arch_Cajas;
procedure Estadisticas_cajas;
procedure estadisticas_clientes;
procedure Estadisticas_generales (var arch_clientes :contador_gral;
                var f_max_est :frecuen; var aux : aoperentero);
procedure generales2;
{**************************************************************}
{**************************************************************}
IMPLEMENTATION
{**************************************************************}
{Los siguientes dos procedimientos nos permitan pasar a trabajar de modo texto
a modo grafico y a la inversa.}
Procedure Inicia_G;
Var i:Integer;
   grDriver : Integer;
   grMode   : Integer;
   ErrCode  : Integer;
Begin
   if allocatescreen(1,80,25) then begin
```

```
    SelectScreen(1); CopyScreen(0);
   end;
  grDriver := Detect;
  InitGraph(grDriver,grMode,'');
  ErrCode := GraphResult;
  if ErrCode  grOk then
   WriteLn('*** Error al inicializar la parte Gráfica ***',
       GraphErrorMsg(ErrCode));
end;
{*************************************************************}
Procedure Termina_G;
Begin
   CloseGraph;
   InTextMode: =TRUE;
   RestoreCrtMode;
   SelectScreen(0); CopyScreen(1);
   DeallocateScreen(1);
   HiddenCursor;
End; {Termina_G}
{*************************************************************}
procedure maximo {(var x:frecuen;i:byte;var ym:integer)};
var k  : integer;
begin
 ym: = 0;
 for k: =  1 to 5 do
  if ym < x[i,k] then ym: = x[i,k];
end;
{*************************************************************}
Procedimiento para presentar las gráficas de barras.
{*************************************************************}
procedure barras {(var x : frecuen; i : byte;s1,s2:string)};
const
  Gray50 : FillPatternType = ($AA, $55, $AA, $55, $AA, $55, $AA, $55);
var
  Ymax, incx,
  Ey,y1,j,y2,y22,y3,
  x1,x2,x3       : Integer;
  s              : string;
  tot            : word;
  aux            : real;
  OldPattern     : FillPatternType;
```

```
  cch          : char;
begin
   STR(I,s);
   SetTextStyle(triplexfont, VertDir,5);
   OutTextXY(9,120,'frecuencia');
   OutTextXY(325,130,'frecuencia acumulada');
 s1:=s1+ s;
 SetTextStyle(DefaultFont, HorizDir,1);
   OutTextXY(150-TextHeight(s1)div 2, 3,S1);
 Ymax:=300;
 Ey:=350;
 x1:=30;
 x2:=345;
 line(x1,35,x1,Ey);    {eje y 1a. graf.}
 line(x1-15,Ey,305,Ey); {eje x 1. graf.}
 line(x2,35,x2,Ey);    {eje y 2a. graf.}
 line(x2-15,Ey,639,Ey); {eje x 2a. graf.}
 incx:=40;
 OutTextXY(240-TextHeight(s2)div 2,378 ,S2);
{  OutTextXY(440-TextHeight(s2)div 2,378 ,S2);}
 maximo(x,i,y1);
 SetTextStyle(DefaultFont, HorizDir,2);
 ymax:=9; y3:=x[i,6]-x[i,7]; tot:=0;
  for j:=1 to 5 do
       tot:=tot+x[i,j];
     aux:=0;
     str(tot,s);
     OutTextXY(170,430,'total de registros '+s);
 SetTextStyle(DefaultFont, HorizDir,1);
 for j:=1 to 5 do
   begin
     str(x[i,j],s); inc(Ymax);
     SetFillStyle(j,Ymax);
     SetColor(Ymax);
     y2:=Ey-trunc((x[i,j]/y1)*300);
     OutTextXY(11,y2,s);
     aux:=aux+(x[i,j]/tot)*100;
     str(trunc(aux),s);
     y22:=Ey-trunc(aux*3);
     OutTextXY(x2-19,y22,s);
     x3:=x1+j*incx;
```

```
      Bar3D(x3 + 9, y2, x3 + incx,Ey, Ymax, TopOn);
      str(y3,s);
      OutTextXY(x3 + 9-textHeight(s),Ey + 9,s);
      x3: = x2 + j*incx;
      Bar3D(x3 + 9, y22, x3 + incx,Ey, Ymax, TopOn);
      OutTextXY(x3 + 9-textHeight(s),Ey + 9,s);
      inc(y3,x[i,7]);
    end;
      x3: = x1 + 6*incx;
      str(y3,s);
      OutTextXY(x3 + 9-textHeight(s),Ey + 9,s);
      x3: = x2 + 6*incx;
      OutTextXY(x3 + 9-textHeight(s),Ey + 9,s);
      Cch : = ReadKey;
end;
{*************************************************************}
procedure abre_arch_clientes;
var
s  : string;
i  : integer;
begin
   for i: = 1 to  parametro.nume_tipos_clientes do
    begin
      str(i,s);
      assign(F_C[i],'clien' + s + '.dat');
      reset (F_C[i]);
    end;
    assign(arch_col,'prom_col.dat');
    reset(arch_col);
end;
{*************************************************************}
procedure abre_arch_oper;
var
  s   : string;
  i   : integer;
begin
   for i: = 1 to parametro.nume_de_operaciones do
   begin
      str(i,s);
      assign(F_O[i],'oper' + s + '.dat');
      reset(F_O[i]);
   end;
```

```
end;
{*************************************************************}
procedure abre_Arch_Cajas;
var s : string;
  i : integer;
begin
  assign(F_Ca,'cajas.dat');
  reset(F_Ca);
end;
{*************************************************************}
procedure Estadisticas_cajas;
var
i,j  : integer;
s2,s1  : string;
aux  : real;
k   : word;
begin
 abre_Arch_Cajas;
 k:=filesize(F_Ca);
 k:=k-2*parametro.nume_tipos_clientes;
 seek(F_Ca,k);
  for i:=1 to parametro.nume_tipos_clientes do
   for j:=1 to 5 do
    begin
     f_min_est[i,j]:=0;
     f_cli_est[i,j]:=0;
     f_ocup_est[i,j]:=0;
     f_max_est[i,j]:=0;
    end;
  for i:=1 to parametro.nume_tipos_clientes do
   begin
    read(F_Ca,E_caja); read(F_Ca,E_caj1);
    with E_caj1 do
     begin
      aux:= E_caja.t_max-t_max;
      aux:=aux/5; aux:=trunc(aux+1);
      f_max_est[i,7]:=trunc(aux);
      f_max_est[i,6]:=t_max;
      for j:=1 to 5 do
       t_max_frec[i,j]:=t_max+trunc(aux*(j));
      aux:= E_caja.t_min-t_min;
      aux:=aux/5; aux:=trunc(aux+1);
```

```
    f_min_est[i,6]: = t_min;
    f_min_est[i,7]: = trunc(aux);
    for j: = 1 to 5 do
     t_min_frec[i,j]: = t_min + trunc(aux*(j));
    aux: = E_caja.cliente_atendidos-cliente_atendidos;
    aux: = aux/5; aux: = trunc(aux + 1);
    f_cli_est[i,6]: = cliente_atendidos;
    f_cli_est[i,7]: = trunc(aux);
    for j: = 1 to 5 do
     clientes_frec[i,j]: = cliente_atendidos + trunc(aux*(j));
    aux: = E_caja.t_ocupa-t_ocupa;
    aux: = (aux/5)*100; aux: = trunc(aux + 1);
    f_ocup_est[i,6]: = trunc(E_caja.t_ocupa*100-(aux*5));
    f_ocup_est[i,7]: = trunc(aux);
for j: = 1 to 5 do ocupacion_frec[i,j]: = trunc(E_caja.t_ocupa*100-(aux*(5-j)));
   end; {del with}
  end; {for i}
 seek(F_ca,0); i: = 0;
 while i < k do
 begin
  read(F_Ca,E_caja);  inc(i);
  with E_caja do
   begin
      if numero < = parametro.cambio_de_caja[1] then j: = 1 else
      if numero < = parametro.cambio_de_caja[2] then j: = 2
      else j: = 3;
      if t_max < t_max_frec[j,1] then inc(f_max_est[j,1]) else
      if t_max < t_max_frec[j,2] then inc(f_max_est[j,2]) else
      if t_max < t_max_frec[j,3] then inc(f_max_est[j,3]) else
      if t_max < t_max_frec[j,4] then inc(f_max_est[j,4]) else
       inc(f_max_est[j,5]);
      if t_min < t_min_frec[j,1] then inc(f_min_est[j,1]) else
      if t_min < t_min_frec[j,2] then inc(f_min_est[j,2]) else
      if t_min < t_min_frec[j,3] then inc(f_min_est[j,3]) else
      if t_min < t_min_frec[j,4] then inc(f_min_est[j,4]) else
       inc(f_min_est[j,5]);
      if cliente_atendidos < clientes_frec[j,1] then inc(f_cli_est[j,1]) else
      if cliente_atendidos < clientes_frec[j,2] then inc(f_cli_est[j,2]) else
      if cliente_atendidos < clientes_frec[j,3] then inc(f_cli_est[j,3]) else
      if cliente_atendidos < clientes_frec[j,4] then inc(f_cli_est[j,4]) else
       inc(f_cli_est[j,5]);
      aux: = t_ocupa*100;
```

```
        if aux < ocupacion_frec[j,1] then inc(f_ocup_est[j,1]) else
        if aux < ucupacion_frec[j,2] then inc(f_ocup_est[j,2]) else
        if aux < ocupacion_frec[j,3] then inc(f_ocup_est[j,3]) else
        if aux < ocupacion_frec[j,4] then inc(f_ocup_est[j,4]) else
         inc(f_ocup_est[j,5]);
    end;{del with}
  end;
       inicia_g;
      for i: = 1 to parametro.nume_tipos_clientes do
       begin
        s1: = 'CLIENTES ATENDIDOS EN CAJAS, TIPO ';
        s2: = 'No. de Clientes ';
        barras (f_cli_est,i,s1,s2);
        ClearViewPort;
        s1: = '% DE OCUPACION DE CAJAS, CLIENTES TIPO ';
        s2: = '% de ocupacion ';
        barras (f_ocup_est,i,s1,s2);
        ClearViewPort;
        s1: = ' TIEMPOS MINIMOS EN CAJA, CLIENTES TIPO ';
        s2: = ' tiempo en segundos ';
        barras (f_min_est,i,s1,s2);
        ClearViewPort;
        s1: = 'TIEMPOS MAXIMOS EN CAJA, CLIENTES TIPO ';
        s2: = ' tiempo en segundos ';
        barras (f_max_est,i,s1,s2);
        ClearViewPort;
       end; {del for}
       termina_g;
end;
{*************************************************************}
Procesa los archivos referidos a clientes.
{*************************************************************}
procedure estadisticas_clientes;
var
i,j    : integer;
W_oper  : windowptr;
s2,s1   : string;
aux     : real;
med,med1 : aoperentero;
k       : word;
begin
   If not MakeWindow(W_oper,1,1,80,10,TRUE,TRUE,FALSE,TextAttr,
```

```
            TextAttr,TextAttr,'') Then ErrorMem;
 If Not DisplayWindow(W_oper) then ErrorMem;
 presenta_titulo('PROCESANDO ARCHIVOS DE CLIENTES','',
            arriba,C_Msg);
abre_arch_clientes;
  k:=filesize(arch_col);
  for i:= 1 to parametro.nume_tipos_clientes do
   begin
      med[i]:=0;
      med1[i]:=0;
   end;
  while not eof(arch_col) do
  begin
     read(arch_col,tiempo_total_cola);
     read(arch_col,clientes_cola_cero);
     {calculo de tiempo medio de cola en minutos}
     for i:= 1 to parametro.nume_tipos_clientes do
      begin
        med[i]:=med[i]+ 2*(tiempo_total_cola[i]/k);
        med1[i]:=med1[i]+ 120*(clientes_cola_cero[i]/k);
      end;
  end;
for i:=1 to parametro.nume_tipos_clientes do
 begin
   k:=filesize(F_C[i]);
   k:=k-2;
   seek(F_C[i],k);
  {inicializo en cero, la tabla de frecuencias}
  for j:=1 to 5 do
   begin
    f_cli_est[i,j]:=0; {tiempo de atencion}
    f_ocup_est[i,j]:=0;{tiempo de cola}
   end;
        aux:=2*med1[i];
      aux:=aux/5; aux:=trunc(aux+1);
      f_cli_est[i,7]:=trunc(aux);
      f_cli_est[i,6]:=trunc(aux);
      for j:=1 to 5 do
        clientes_frec[i,j]:=f_cli_est[i,6]+trunc(aux*j);
      aux:=2*med[i];
      aux:=aux/5; aux:=trunc(aux+1);
      f_ocup_est[i,6]:=trunc(aux);
```

```
      f_ocup_est[i,7]:=trunc(aux);
      for j:=1 to 5 do
      ocupacion_frec[i,j]:=f_ocup_est[i,6]+trunc(aux*(j));
   end; {for i}
 {el siguiente for levanta estadísticas}
for j:=1 to parametro.nume_tipos_clientes do
begin
   k:=filesize(F_C[j]);
   k:=k-2;
   seek(F_C[j],0); i:=0;
  while i<k do
  begin
    read(F_C[j],E_clie);   inc(i);
    with E_clie do
     begin
       if tiempo_atencion<clientes_frec[j,1] then inc(f_cli_est[j,1]) else
       if tiempo_atencion<clientes_frec[j,2] then inc(f_cli_est[j,2]) else
       if tiempo_atencion<clientes_frec[j,3] then inc(f_cli_est[j,3]) else
       if tiempo_atencion<clientes_frec[j,4] then inc(f_cli_est[j,4]) else
        inc(f_cli_est[j,5]);
       if trunc(tiempo_de_cola/60)  ocupacion_frec[j,1] then
                          inc(f_ocup_est[j,1]) else
       if trunc(tiempo_de_cola/60)  ocupacion_frec[j,2] then
                          inc(f_ocup_est[j,2]) else
       if trunc(tiempo_de_cola/60)  ocupacion_frec[j,3] then
                          inc(f_ocup_est[j,3]) else
       if trunc(tiempo_de_cola/60)  ocupacion_frec[j,4] then
                          inc(f_ocup_est[j,4]) else
        inc(f_ocup_est[j,5]);
     end;{del with}
  end;{del while}
end;{del for que levanta estadisticas}
 inicia_g;
    for i:=1 to parametro.nume_tipos_clientes do
     begin
       s1:= 'TIEMPO DE ATENCION EN CAJAS, CLIENTES TIPO ';
       s2:=' tiempo en segundos ';
       barras (f_cli_est,i,s1,s2);
       ClearViewPort;
       s1:='TIEMPO EN COLA, CLIENTES TIPO ';
       s2:=' tiempo en minutos ';
       barras (f_ocup_est,i,s1,s2);
```

```
        ClearViewPort;
      end; {del for}
   termina_g;
    apaga_mensaje(W_oper);
  end;{de estadisticas_clientes}
  procedure abre_arch_generales;
  begin
    assign(arch_clientes,'clientes.dat');
    assign(arch_No_oper,'No_oper.dat');
    assign(arch_colas,'colas.dat');
    reset(arch_clientes);
    reset(arch_No_oper);
    reset(arch_colas);
  end;
  {*************************************************************}
  Porcesa los archivos de generales, clientes atendidos por día, tiempos medios en
cola, etc.
  {*************************************************************}
  procedure Estadisticas_generales(var arch_clientes :contador_gral;
                    var f_max_est :frecuen; var aux : aoperentero);
  var
  i,j  : integer;
  s2,s1   : string;
  desv,aux1  : array [1..3] of real;
  k    : word;
  begin
   k:=filesize(arch_clientes);
   { Calculo de medias }
   for i:=1 to 3 do begin aux[i]:=0; desv[i]:=0; end;
   while not eof(arch_clientes) do
   begin
    read(arch_clientes,cuenta_cliente);
    for i:=1 to parametro.nume_tipos_clientes do
        aux[i]:=aux[i]+cuenta_cliente[i]/k;
   end;
   seek(arch_clientes,0);
   {calculo desviación}
   while not eof(arch_clientes) do
   begin
    read(arch_clientes,cuenta_cliente);
    for i:=1 to parametro.nume_tipos_clientes do
        desv[i]:=desv[i]+Sqr(aux[i]-cuenta_cliente[i])/(k-1);
```

```
 end;
 for i:=1 to parametro.nume_tipos_clientes do
 begin
   desv[i]:=sqrt(desv[i]);
   aux1[i]:=2*desv[i];
  for j:=1 to 5 do
   begin
    f_max_est[i,j]:=0;
   end;
 end;
 for i:=1 to parametro.nume_tipos_clientes do
  begin
     f_max_est[i,6]:=trunc(aux[i]-desv[i]);
     aux1[i]:=aux1[i]/3; aux1[i]:=trunc(aux1[i]+1);
     f_max_est[i,7]:=trunc(aux1[i]);
     for j:=1 to 5 do
      t_max_frec[i,j]:=f_max_est[i,6]+trunc(aux1[i]*(j-1));
  end; {for i}
 seek(arch_clientes,0);
 { levanto frecuencias}
 while not eof(arch_clientes) do
 begin
  read(arch_clientes,cuenta_cliente);
  for i:=1 to parametro.nume_tipos_clientes do
      if cuenta_cliente[i]  <t_max_frec[i,1] then inc(f_max_est[i,1]) else
      if cuenta_cliente[i]  <t_max_frec[i,2] then inc(f_max_est[i,2]) else
      if cuenta_cliente[i]  <t_max_frec[i,3] then inc(f_max_est[i,3]) else
      if cuenta_cliente[i]  <t_max_frec[i,4] then inc(f_max_est[i,4]) else
       inc(f_max_est[i,5]);
 end; {while}
end;
{*************************************************************}
Presenta en pantalla las graficas de estadísticas señaladas como generales.
{*************************************************************}
procedure generales2;
var
aux  : aoperentero;
i  : integer;
s1,s2  : string;
begin
     abre_Arch_generales;
```

```
    Estadisticas_generales(arch_clientes, f_max_est,aux);
     inicia_g;
    for i: = 1 to parametro.nume_tipos_clientes do
     begin
      s1: = 'CLIENTES ATENDIDOS POR DIA, TIPO ';
      str(aux[i]:6:2,s2);
      s2: = ' No. Promedio de Clientes ' + s2;
      barras (f_max_est,i,s1,s2);
      ClearViewPort;
     end; {del for}
    Estadisticas_generales(arch_colas, f_max_est,aux);
    for i: = 1 to parametro.nume_tipos_clientes do
     begin
      s1: = 'COLAS CLIENTES, TIPO ';
      str(aux[i]:6:2,s2);
      s2: = ' No. Promedio de Cola ' + s2;
      barras (f_max_est,i,s1,s2);
      ClearViewPort;
     end; {del for}
     termina_g;
end;
END.
{*************************************************************}
```

UNIDAD TYPOSTESIS.

Esta unidad esta conformada por tres secciones, constantes, typos y variables. Las constantes son las que establecen las limitaciones , que en este caso, se le impusierón al sistema, tales como números máximo de cajas, número máximo de tipos de clientes, número máximo de operaciones a realizar, duración de un día de simulación (5 hrs, en segundos, que fué la unidad de tiempo que se eligio). Como se verá más adelante algunos de los typos que a continuación definiremos, estan limitados por estas constantes. Lo que facilita incorporar futuras modificaciones al sistema, facilmente.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}

unit typotesi;
interface
const
num_de_oper  = 10;{número máximo de operaciones por cliente}
   nitll      = 10; {número máximo de variaciones de
                 tiempos medios de llegadas}
   n_d_cajas  = 10; {número máximo de cajas}
   num_de_clientes = 3; {tipos de clientes distintos}
   dia_seg    = 18000; {5 hrs. en segundos}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Hay dos tipos, que son entidades dinámicas, denominadas apuntadores, que nos serviran para la contrucción de lineas de espera de clientes y una cola de eventos futuros.

El primero de ellos Evento, nos indica la hora en que va a ocurrir y el tipo de evento, este puede ser liberar la caja indicada, si el valor correspondiente a "tipo" esta entre 1 y 10, en caso de ser cero, se produce una nueva llegada al banco.

Despúes de definir evento, se define un registro (oper1), de dos campos, en el primero para asignar el tiempo de servicio de una operación y el segundo asignarle un valor por esa operación, despues viene un vector de ese tipo de registros (vector_de_oper1).

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
type
```

```
Evento    = ^even1;
Even1 '   = record
             tiempo : word;
             tipo   : byte;
             siguiente:evento;
            end;
 oper1    = record
          tiempo, valor  : integer;
          end;
 vector_de_oper1   = array[1..num_de_oper] of oper1;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

A continuación se define, La segunda entidad dinámica del sistema, denominada cliente, que nos permite asignarle a cada cliente: el tipo o categoria de cliente, la hora de llegada al banco, la hora en que es atendido, la hora en que avandona las instalaciones y un vector donde se le asignan los valores correspondientes a cada operación que ejecuta.

El siguiente vector de clientes nos servira para construir las colas, de cada tipo de clientes.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  cliente  = ^xcliente;
  xcliente = record
          tipo        : byte;
          nacimiento,  { hora de ingreso al banco}
          despacho,    { hora de ser atendido}
          salida       : word;
          operacion    : vector_de_oper1;
          sig          : cliente;
         end;
 vector_cola = array [1..num_de_clientes] of cliente;

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

El siguiente registro, **caja1** nos servirá para representar una caja, donde almacenaremos algunos parámetros que obtendremos como resultados de la simulación. Los campos desocupada de tipo booleano indicará en que estado se encuentra la caja (ocupada o desocupada), t_de_ocupación registrará el tiempo total de ocupación de la caja, t_de_liberar indicará la hora en que un cliente términa de ser atendido y por consiguiente se liberará la caja, t_max_de_atención guarda el tiempo máximo de atención a un cliente en esa caja, t_minimo_de_atención al-

macena el tiempo mínimo, clientes-atendidos es un contador de clientes servidos. El siguiente tipo es un vector de cajas, que nos facilitará el manejo de las cajas.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  caja1    = record
         desocupada    : boolean;
         t_de_ocupacion : real;
         t_de_liberar   : word;
         t_max_de_atencion,
         t_minimo_de_atencion,
         clientes_atendidos  : integer;
        end;
 vector_caja = array[1..n_d_cajas] of caja1;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

El siguiente bloque de tipos nos servirán para definir el registro de parámetros, que contiene la totalidad de parámetros que requiere la simulación.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}

  camb_caja        = array [1..num_de_clientes-1] of byte;
  tiempo_de_llegada = record
             hasta : integer;
             c_p_m : real; {clientes por min}
            end;
  vector_tiempo_medio_de_llegadas  =array[1..nitll] of tiempo_de_llegada;
  aoperentero  = array[1..num_de_clientes] of real;
  oper       = record
          nombre    : string20;
          t_m_s,{tiempo medio de servicio}
          m_t_m,{desviación del tiempo de servicio}
          dis_t_m_s,  {tipo de distribución}
          valor_medio,
          dis_v_m,
          porcentaje : aoperentero;
          end; {de oper}
  vector_de_operaciones     = array[1..num_de_oper] of oper;
  parametros = record
      cambio_de_caja,
      Por_cliente      : camb_caja;
      nume_de_cambios_de_llegada,
```

```
nume_de_operaciones, nume_de_cajas,
nume_tipos_clientes,
banqueros1,
banqueros2        : byte;
operacion         : vector_de_operaciones;
t_de_llegadas     : vector_tiempo_medio_de_llegadas;
end; {record de parámetros}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Los siguientes tipos, se diseñarón con la finalidad de seccionar la información de la simulación en varios archivos de trabajo, que posteriormente a la simulación son procesados en el modulo de estadísticas. El registro Caja_est, almacena el número de la caja, el tiempo máximo y el mínimo en atender a un cliente, los clientes atendidos en un día y finalmente el tiempo total que estuvo ocupada. El registro Cliente_est, almacena el tipo de cliente, tiempo de atención y el tiempo de cola, el vector V_E_C nos permite almacenar la información de todos los tipos de clientes. Finalmente el registro Est_oper indica el tipo de cliente el tiempo y el valor de una de las operaciones, y el vector V_E_O nos facilita guardar esa información de todas las operaciones que se realicen.

Los últimos dos arreglos, son contadores de clientes por tipo y de número de cada operación realizada por tipo de cliente.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}

Caja_est   = record
        numero,
        t_max,
        t_min,
        cliente_atendidos : integer;
        t_ocupa           : real;
       end;
cliente_est = record
         tipo            : byte;
         tiempo_atencion,
         tiempo_de_cola  : word;
        end;
Est_oper   = record
        tipo             : byte;
        tiempo,
        valor            : integer;
       end;
```

```
    V_E_O      = array[1..num_de_oper] of Est_oper;
    V_E_C      = array[1..num_de_clientes] of cliente_est;
    cuent_clie = array[1..num_de_clientes] of word;
  cuenta_oper = array[1..num_de_clientes,1..num_de_oper] of word;
```

{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}

El siguiente bloque, define los archivos en que abra de guardarse la información recabada.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}

```
    arch_est_oper      = file of Est_oper;
    arch_est_cliente   = file of cliente_est;
    arch_est_caja      = file of Caja_est;
    V_arch_est_oper    = array [1..num_de_oper] of arch_est_oper;
    V_arch_est_cliente = array [1..num_de_clientes] of arch_est_cliente;
    contador_gral      = file of cuent_clie;
```

{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}

El tipo frecuen, se utiliza para la elaboración de tablas de frecuencia, que se utiliza en el módulo de estadiísticas. Los tipos de vector_entero y vector real, se utiliza en el modulo de captura de datos para transformar a segundos los tiempos de cambio de tiempos medios de llegada.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}

```
    frecuen      = array[1..3,1..7] of word;
    string20     = string[20];
    vector_entero = array [0..18] of integer;
    vector_real  = array [1..18] of real;
```

{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}

Las siguientes variables, son comunes a los distintos procedimientos que integran las unidades.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}

```
  var
    n_de_c,  {número de cajas}
    n_de_c_d, {número de clientes diferentes}
    n_d_o_d,  {número de operaciones diferentes}
    n_d_v_t {número de variaciones de tiempos de llegada} : byte;
    operacion              : vector_de_operaciones;
    parametro              : parametros;
    archivo                : file of parametros;
    C_Msg,C_Num,C_Lee,C_Tit    : byte;
    Ult_par      : file of parametros;
    arch_colas,
    arch_clientes  : contador_gral;
    arch_No_oper   : file of cuenta_oper;
```

```
  E_caja,
  E_caj1        : Caja_est;
  Ma_clie,
  Mi_clie       : V_E_C;
  E_clie        : cliente_est;
  E_oper        : Est_oper;
  Ma_e, Mi_e    : V_E_O;
  F_Ca          : arch_est_caja;
  F_O           : V_arch_est_oper;
  F_C           : V_arch_est_cliente;
  st     : byte; {indicador de tiempo medio de llegada}
  caja          : vector_caja;
  nuevo_evento,
  sig_evento,
  cola_evento,
  cabeza_evento,
  fin_evento    : evento;
  t_reloj,siguiente       : word;
  cola_muertos,
  cabeza_muertos,
  cleo          : cliente;
  cabeza,cola   : vector_cola;
  bandera,banda : boolean;
  ef            : integer;
  esta          : array[1..6] of est_oper;
  t_max_frec,t_min_frec,
  clientes_frec,ocupacion_frec,
  f_min_est,f_cli_est,
  f_ocup_est,f_max_est        : frecuen;
  cuenta_cliente,cola1        : cuent_clie;
  cuen_oper                   : cuenta_oper;
implementation

end.
```

PROGRAMA PRINCIPAL BANCOSIM.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}

Los procedimientos Presenta_fondo, termina e inicia_global, se integran en el procedimiento Presentación de entrada, que pretende ser un llamado de atención agradable hacia el usuario, cuando este entra al sistema.

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}

```
Uses Dos, TpCrt, TPString, TPWindow, TpScreen, Sounds, Gun_Ent,
   ventanas, PARA2,SIMULA,typotesi,estadis;
Var
  Base    :Pointer;
  Raiz    :Apt_Encabezado;
  SubMenu :Apt_Opcion;
 {*************************************************************}
  {E N T R A D A   D E   P R E S E N T A C I O N}
 {*************************************************************}
 Procedure Presenta_Fondo;
 Begin {Presenta_Fondo}
   ClrScr;
   SetFrameChars(",",",",",");
FrameWindow(1,1,80,25,TextAttr, TextAttr,' SIMULACION BANCARIA');
   Window(1,1,80,25);
 End; {Presenta_Fondo}
{*****************************************************************}
 Procedure Termina;
 Begin
   NormalCursor;
   TextColor(White); TextBackGround(Blue);
   ClrScr;
 End; {Termina}
{*****************************************************************}
Procedure Inicia_Global;
Begin
   ReInitCrt; ClrScr;
   SoundFlagW:=False; Shadow:=true; Explode:=true;
   BiosScroll:=FALSE;
   TextColor(White); TextBackGround(Blue);
```

```
End; {Inicia_Global}
{**********************************************************}
Procedure Presentacion_de_Entrada;
Begin
  {PARTE EN MODO DE GRAFICAS}
  Inicia_Entrada;
  Linea_Entrada('POSGRADO DE INGENIERIA');
  Linea_Entrada('');
  Linea_Entrada('SIMULACION');
  Linea_Entrada('BANCARIA');
  Linea_Entrada('');
  Linea_Entrada('INVESTIGACION');
  Linea_Entrada('DE');
  Linea_Entrada('OPERACIONES');
  Muestra_Entrada;
  {PARTE EN MODO TEXTO}
  Inicia_Letras;
  Presenta_Fondo;
  Lee_Letras;
  Window(2,2,79,24); {Para que no afecte el marco}
End; {Presentacion_de_Entrada}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

El siguiente procedimiento, define los distintos menus que integran el sistema, el primero es el menú principal y despues en orden los submenús correspondientes a cada opción.

```
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
{**********************************************************}
  {EJECUCION DEL PROGRAMA}
Procedure Menus;
Begin
   Presenta_Fondo;
   Crea_Pantalla(Raiz,NIL,'OPCIONES DEL SISTEMA');
   Inserta(Raiz,'MODIFICACION DE PARAMETROS');
   Inserta(Raiz,'RESULTADOS DE LA SIMULACION');
   Inserta(Raiz,'REALIZAR UNA SIMULACION');
   Inserta(Raiz,'SALIR DEL SISTEMA');
   Crea_Ventana(Raiz,2);
   SubMenu:=Raiz^.Primero;
   With SubMenu^ do Begin
   Crea_Pantalla(Cabeza,Raiz,'MODIFICACION DE PARAMETROS');
      Inserta(Cabeza,' GENERALES (TODOS) ');
      Inserta(Cabeza,' TIPOS DE CLIENTES ');
```

```
        Inserta(Cabeza,' OPERACIONES   ');
        Inserta(Cabeza,' TIEMPOS DE LLEGADA  ');
        Inserta(Cabeza,' SELECCION DE DISCO  ');
        Inserta(Cabeza,' SALIR  ');
        Crea_Ventana(Cabeza,25);
     End; {With}
     SubMenu:=SubMenu^.Sig;
     With SubMenu^ do Begin
        Crea_Pantalla(Cabeza,Raiz,'CORRIDA DE SIMULACION');
        Inserta(Cabeza,'TABLAS DE FRECUENCIA DE CAJAS');
        Inserta(Cabeza,'TABLAS DE FRECUENCIAS DE CLIENTES');
        Inserta(Cabeza,'TABLAS DE FRECUENCIAS GENERALES');
        Inserta(Cabeza,'SALIR');
        Crea_Ventana(Cabeza,25);
     End; {With}
     Submenu:=SubMenu^.Sig;
     With SubMenu^ do Begin
        Crea_Pantalla(Cabeza,Raiz,'REALIZAR UNA SIMULACION');
        Inserta(Cabeza,'INICIO DE LA SIMULACION');
        Inserta(Cabeza,'SALIR');
        Crea_Ventana(Cabeza,25);
     End; {SubMenu}
  End; {Menus}
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

El procedimiento "aplicando" a partir de dos variables nivel1 y nivel2, que serán producto de la selección hecha por el usuario en el menú principal (nivel_1) y posteriormente en el submenú correspondiente (nivel_2) indica el procedimiento a ejecutar. Este es el procedimiento que nos permite ver la integración de todo el sistema. Que se retoma en el procedimiento Aplicación Menus y finalmente en lo que es el programa principal.

```
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  {****************************************************************}
  {P R O C E D I M I E N T O S   D E   E J E C U C I O N}
  {****************************************************************}
  Procedure Aplicando(Var Nivel_1, Nivel_2: Byte;Mensaje:String);
  Var Trabajo:WindowPtr; Color_Mensaje: Byte;
  Begin
   Color_Mensaje:=PC(Yellow,Blink+Green);
   Case Nivel_1 of
     1: begin
        abre_archivo(bandera);
        if bandera then
```

```
        begin
          read(archivo,parametro);
          Case Nivel_2 of
             1: Establece_parametros(parametro);
             2: Nuevos_tipos_de_cliente;
             3: Nuevos_tipos_de_operaciones;
             4: Nuevos_tiempos;
             5: Seleccion_de_disco(parametro);
          End; {Case}
        End; {del if}
      End;
    2:BEGIN assign(ult_par,'ultimo.dat');
        reset(ult_par);
        read(ult_par,parametro);
        case nivel_2 of
          1 : estadisticas_cajas;
          2 : estadisticas_clientes;
          3 : generales2;
        end;
     END;
    3:BEGIN if Nivel_2 = 1 then corrida  END;
 End; {Nivel_1}
End; {Aplicando}
{*****************************************************************}
Procedure Aplicacion_Menus;
Var
  Resultado, Nuevo_Resultado       :Apt_Opcion;
  Num_Primero, Num_Segundo         :Byte;
  Salida1, Salida2                 :Boolean;
  V                                :WindowPtr;
Begin
   Salida1: = False;

   Repeat
      Presenta(Raiz,Resultado,Num_Primero);
      If ResultadoRaiz^.Ultimo then Begin
        Salida2: = False;
        Repeat
           Presenta(Resultado^.Cabeza,Nuevo_Resultado,Num_Segundo);
           If Nuevo_ResultadoResultado^.Cabeza^.Ultimo then
             Aplicando(Num_Primero,Num_Segundo,
```

```
                    Trim(Resultado^.Titulo)+' - '
                    +Trim(Nuevo_Resultado^.Titulo))
            else Salida2:=True;
          Until Salida2;
          V:=EraseTopWindow;
        end Else Salida1:=true;
      Until Salida1;
      V:=EraseTopWindow;
  End; {Aplicacion_Menus}
  {**********************************************************}
```

El siguiente es el programa principal, las primeras tres instrucciones, corresponden al llamado de el saludo del programa, la instrucción menus, llama al procedimiento de ese nombre, que define los menus del sistema, mientras que aplicación_menus, es la parte iterativa del sistema.

```
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}}}}}}}}}}}}
  Begin {Programa Principal}
    Inicia_Global;
    banda:=false;
    If ParamCount=0 then Presentacion_de_Entrada;
    Menus;
    Aplicacion_Menus;
    Termina;
  End.
```

CONCLUSIONES.

El uso creativo de la imaginación no es sólo la fuente de la inspiración poética y artística, sino también de los descubrimientos científicos; en efecto, la imaginación suministra el impulso inicial en todo progreso. El poder creador de la imaginación ha inspirado, sin excepción, los grandes descubrimientos de la ciencia.

Sir William Huggins, astrónomo inglés (1824-1909).

La aplicación de la simulación digital en el análisis de sistemas, es un problema práctico, el desarrollo actual de la computación, permite la elaboración de modelos de sistema con un alto acercamiento a la realidad. Sin embargo, la parte más difícil en la simulación sigue siendo la definición del sistema y la fase de formulación del modelo.

La habilidad y cuidado en el diseño del modelo, así como su constitución modular adecuada, nos permitirá mejorarlo y corregirlo en etapas finales, conforme se adquiere experiencia en el sistema, mientras más errores cometamos inicialmente, más difícil y costoso será efectuar cualquier modificación que se requiera hacer en el modelo.

El desarrollo de sistemas de simulación en este momento, nos proporciona una rica experiencia en el modelado, lo que nos permitirá cada vez, simular sistemas más complejos. La utilización de lenguajes estructurados, tales como el C + +, y el turbo pascal nos permite ir construyendo un lenguaje propio, donde se manifiesta la experiencia adquirida, dicho de otra forma, nos permite reutilizar procedimientos o módulos completos de simulaciones hechas en el pasado, en la realización de nuevas. El uso de estos lenguajes en constante evolución, nos permite incorporar los adelantos alcanzado en ellos, por ejemplo la utilización de listas dinámicas mediante la incorporación de apuntadores.

Por estas razones el sistema "BANCOSIM" fue desarrollado en turbo pascal versión 5.5, y como puede apreciarse en el manual técnico, el desarrollo com-

putacional es modular.

El sistema "BANCOSIM" maneja un total de 230 parámetros sobre los cuales el usuario puede interaccionar, ya sea modificandolos y/o reduciendolos.

Se realizaron pruebas de compilación para determinar el número de parámetros que podía tolerar el sistema, verificando que soporta 150 operaciones diferentes y 180 cajas de servicio, lo que nos da un poco más de 3000 parámetros del modelo, dicho en otras palabras, con la tecnología disponible en el mercado nacional, podemos desarrollar sistemas con alto grado de complejidad, radicando el problema fundamental en un buen modelado de la realidad más que en los requerimientos tecnológicos.

La principal limitación del sistema presentado, es que el modelado es muy superficial, debido a que no se contó con la participación de especialistas en el movimiento de un sistema bancario, sin embargo el objetivo del sistema es demostrar que podemos desarrollar sistemas amigables para un usuario. Esta limitación nos impidió considerar algunos aspectos que con seguridad influyen en una sucursal bancaria, no todo es servicio en cajas. La simulación puede considerar el papel que juega cada uno de los personajes del sistema como son gerente, subgerente, banqueros de cabecera, etc. Seguramente considerar todos los elementos que intervienen en una sucursal bancaria nos llevaria más tiempo y requeriria la participación de por lo menos un especialista en el movimiento bancario.

El siguiente paso en la simulación digital es la utilización de la programación orientada a objetos, ya que tiene las siguientes características:

-Es compatible con la representación usada en simulación.

-Permite que un programa se escriba centrandose en la descripción del problema, más que en los algoritmos usados para su solución.

-Requiere menos código y mas fácil de modificar.

Algunos de los lenguajes orientados a simulación tienen este enfoque, tales como el simula y Simkit. Sin embargo su disponibilidad es limitada, de ahí que las perspectivas de la utilización de lenguajes como C + + y turbo pascal, cuyo uso está generalizado en las universidades y centros de investigación, tienen más posibilidades de desarrollo que los primeros.

ANEXO

A.1 CONCEPTOS DE PROBABILIDAD USADOS EN SIMULACION.

La mayoría de los modelos de simulación, si es que no todos, son modelos estocásticos. El primer problemas a que nos vamos a enfrentar una vez diseñado el modelo, es la generación o descripción de las variables aleatorias que se vayan a utilizar.

A continuación discutimos este problema, empezando por el caso discreto y después el continuo. En este último, analizamos las distintas posibilidades que se presentan en el modelaje de simulación.

Caso Discreto. Si la variable es discreta y solo puede tomar n valores dados que son xi, $1 < i < n$, cuya probabilidad es pi, sabemos que:

$$\sum_{i=1}^{n} pi = 1$$

Estas probabilidades pueden ser dadas de antemano, o bien determinarse mediante una serie de observaciones, a partir de las cuales se establecen las diferentes probabilidades, como se muestra en el siguiente:

Ejemplo A1: Un puesto ambulante a la entrada de un cine, vende 6 distintos tipos de golosinas, de cien clientes que compraron, se registró el número de golosinas diferentes que compraban, obteniéndose los siguientes resultados:

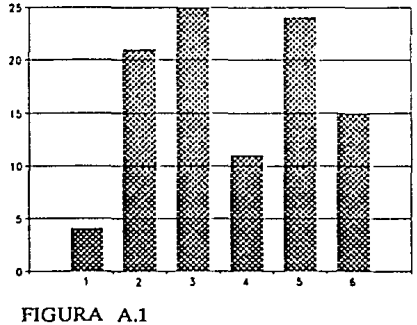
xi	observaciones	probabilidad (pi)	prob.acum.
1	4	0.04	0.04
2	21	0.21	0.25
3	25	0.25	0.50
4	11	0.11	0.61
5	24	0.24	0.85
6	15	0.15	1.00
	100		1.00

La variable estocástica x_i puede tomar los siguientes valores con las siguientes probabilidades:

valor	1	2	3	4	5	6
probabilidad	0.04	0.21	0.25	0.11	0.24	0.15

Para generar números aleatorios con estas probabilidades, podemos hacerlo, considerando la gráfica de la probabilidad acumulada, como se muestra en la figura A1, generamos una sucesión ri, de números aleatorios con distribución uniforme entre cero y uno, y dependiendo el intervalo en que este el número aleatorio será, el valor que le asociemos, por ejemplo:

FRECUENCIA ABSOLUTA



FRECUENCIA ACUMULADA

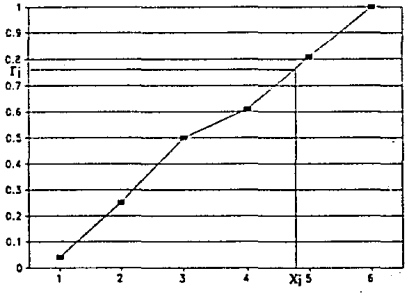


FIGURA A.1

ri	0.274	0.911	0.046	0.4660	0.4976
intervalo	(.25,.50)	(.85,1)	(.04,.25)	(.25,.5)	(.25,.5)
valor de xi	3	6	2	3	3

Caso continuo.- Teóricamente la variable puede tomar un número infinito de valores, y la probabilidad de cada uno de estos valores (teórica) es cero, y la podemos describir a través de una función de distribución de probabilidad (f.d.p.) f(x), que debe cumplir:

i) $f(x) >= 0$ para toda $x \epsilon R$.

y ii) $$\int_{-\infty}^{\infty} f(x)\,dx = 1$$

y la probabilidad de que x tome un valor entre x1 y x2.

$$P(x_1 < x < x_2) = \int_{x2}^{x1} f(x)\,dx$$

Discretización de funciones continuas. Algunas veces resulta conveniente discretizar funciones continuas, en otras ocasiones no tenemos más que esa alternativa, veamos pues los dos casos más frecuentes la figura A2

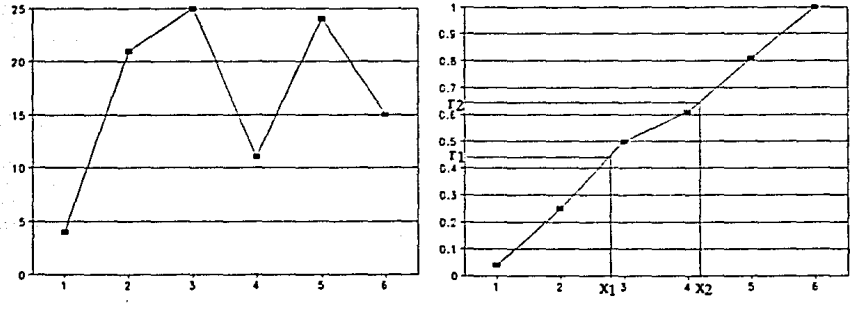


FIGURA A.2

a)La variable aleatoria tiene una función de densidad conocida y continua y = f(x). De donde F(x) la podemos evaluar para algunos xi, esto siempre es posible por métodos numéricos, esto es:

$$F(x) = \int_{-\infty}^{x} f(x)\,dx.$$

Una vez encontrado una sucesión de puntos (xi,F(xi)), los unimos por medio de segmentos de rectas, como se ilustra en la figura 8.

Para generar números aleatorios que sigan esa distribución generamos primero una sucesión de números aleatorio con distribución uniforme ri, cada uno de estos valores, caerá en alguno de los intervalos determinados por F(xi), F(xi + 1), para alguna i, mediante interpolación obtendremos el valor yi buscado. Veamos un ejemplo:

Suponga que la función de densidad esta dada de la siguiente forma:

$$f(x) = \begin{cases} 2(1-x) & \text{si } 0 < x < 1 \\ 0 & \text{en otro caso.} \end{cases}$$

Lo primero que se hace es verificar si efectivamente se trata de una función de densidad, para lo que integramos f(x)

$$\int_{-\infty}^{\infty} f(X)\,dx. = \int_{-\infty}^{0} 0\,dx + \int_{0}^{1} 2(x-1)dx + \int_{1}^{\infty} 0\,dx$$

de donde

$$\int_{-\infty}^{\infty} f(X)\,dx. = 0 + \int_{0}^{1} 2(x-1)dx + 0 = 2x\text{-}x2\Big|_{0}^{2} = 1$$

Como f(x) 0 para toda x, entonces se f es una función de densidad, y además F(x) = 2x-x2 si x(0,1), 1 si x 1 y 0 si x 0, de ahí que la sucesión de xi la debemos de tomar en el intervalo (0,1), esta sucesión no tiene que estar necesariamente a intervalos regulares en x, aquí los tomaremos equidistantes por comodidad, tomemos por ejemplo 10 valores de x y evaluemos para esos valores F(x), como se indica en la siguiente tabla:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
F(x)	0.19	00.36	0.51	0.64	0.75	0.84	0.91	0.96	0.99	1.0

Para fines didácticos, esta división es buena, observe que las variaciones en x, son constantes, mientras que las variaciones de F(x) cada vez son menores, una mejor selección sería empezar con valores más próximos a 0 e ir incrementando la variación de los intervalos.

Una vez que hemos evaluado puntualmente nuestra función, podemos considerarla como lineal entre cada par de puntos, como se muestra en la figura 8, para generar números con esta distribución, generamos una sucesión ri, de números aleatorios con distribución uniforme entre cero y uno, y dependiendo el intervalo en que este, le asignamos un valor interpolando entre los correspondientes valores de las F(x), por ejemplo:

ri	Intervalo de F(x)	Intervalo de x	Interpolación $yi = xi-1 + \frac{(xi-xi-1)*(ri-xi-1)}{F(xi)-F(xi-1)}$	yi
0.517	(.51,.64)	(.3,.4)	.3+[.1/(.64-.51)]*(.517-.51)	0.305
0.240	(.19,.36)	(.1,.2)	.1+[.1/(.36-.19)]*(.240-.19)	0.129
0.459	(.36,.51)	(.2,.3)	.2+[.1/(.51-.36)]*(.459-.36)	0.266
0.305	(.19,.36)	(.1,.2)	.1+[.1/(.36-.19)]*(.305-.19)	0.167
0.035	(0.0,.19)	(0,0.1)	0+[.1/(.19-0.0)]*(.035-0.0)	0.018
0.649	(.64,.75)	(.4,.5)	.4+[.1/(.75-.64)]*(.649-.64)	0.408
0.156	(0.0,.19)	(0,0.1)	0+[.1/(.19-0.0)]*(.156-0.0)	0.082
0.094	(0.0,.19)	(0,0.1)	0+[.1/(.19-0.0)]*(.094-0.0)	0.049
0.216	(.19,.36)	(.1,.2)	.1+[.1/(.36-.19)]*(.216-.19)	0.115
0.910	(.91,.96)	(.7,.8)	.7+[.1/(.96-.91)]*(.910-.91)	0.7

b) No conocemos la expresión analítica de la función de densidad, pero sabemos que es continua y tenemos -o podemos hacer-observaciones de su comportamiento, por ejemplo considere la variable aleatoria de duración de una llamada telefónica, en una caseta pública y suponga que tenemos 100 observaciones, entonces procedemos a agruparlas en intervalos, la siguiente tabla muestra las observaciones ya agrupadas, en la primera columna se indica el intervalo de tiempo considerado, estos intervalos son cerrados superiormente y abiertos en el limite inferior del mismo, la segunda columna muestra el conteo de llamadas que caen en el, la tercera columna corresponde a la frecuencia relativa de las llamadas en el intervalo, es decir número de llamadas del intervalo entre el número total de llamadas registradas y finalmente la última columna corresponde a la frecuencia acumulada hasta el intervalo en cuestión.

Intervalo	Conteo	Frecuencia Relativa	Frecuencia Acumulada
(0,30]	2	0.02	0.02
(30,60]	5	0.05	0.07
(60,90]	7	0.07	0.14
(90,120]	24	0.24	0.38
(120,150]	25	0.25	0.63
(150,180]	18	0.18	0.81
(180,210]	10	0.10	0.91
(210,240]	1	0.01	0.92
(240,270]	3	0.03	0.95
(270,300]	2	0.02	0.97
(300,420]	3	0.03	1.00

De la tabla anterior, podemos graficar la última columna, esto es la frecuencia acumulada, haciendo corresponder el límite superior del intervalo a la frecuencia acumulada del mismo, y posteriormente unimos mediante segmentos de rectas estos puntos, como se muestra en la figura 9, de esta forma para generar números aleatorios con esta distribución, nos encontramos en este momento, en las mismas condiciones que en el caso anterior. De donde basta generar una serie de números aleatorios ri con distribución uniforme entre cero y uno y asociarle una serie yi mediante interpolación.

Hasta ahora hemos reducido el problema de generar números aleatorios con una distribución dada, en forma analítica o empírica a través de series de números aleatorios con distribución uniforme, con seguridad si resolvemos el problema de generar números aleatorios con distribución uniforme, salvaremos el problema general, para cualquier distribución.

A.2 GENERACION DE NUMEROS ALEATORIOS CON DISTRIBUCION UNIFORME, ENTRE CERO Y UNO.

En los ejemplos anteriores, obtuvimos las series de números aleatorios uniformemente distribuidos, de tablas de números con esta distribución que suelen venir al final de los libros de estadística, esta es la primera de las formas de obtener este tipo de números, otra forma es por métodos físicos y una tercera mediante métodos matemáticos.

a) **Métodos Físicos.** Existe una gran variedad de este tipo de métodos para generar números aleatorios con distribución uniforme. La más socorrida de todas es meter en un saco 10 papelitos, o bolas con números del 0 al 9, si deseamos generar números con 3 cifras, entonces sacamos un primer papel o bola observamos el número marcado y lo escribimos, volvemos a meter el papel o la bola al saco, agitamos y repetimos la misma operación, así hasta completar el número de cifras deseadas. Lo importante es resaltar que una vez observado y anotado el número, regresamos el papelito o la bola al saco.

b) **Tablas.** Localice una tabla de números aleatorios uniformemente distribuidos (generalmente los libros de estadística traen una), elija al azar un número y a partir de el, empiece a seleccionar los demás números determinando la forma de irlos seleccionando, esta puede ser hacia abajo, hacia arriba, en diagonal, etc. Lo único que no vale es irlos tomando salteados o sin ningún orden.

c) **Métodos Matemáticos.** Existe una gran variedad de esta forma de generación y dan lugar a generación de estos números, mediante computadora, con frecuencia son llamados también seudoaleatorios, debido a que realmente la sucesión de números a que da lugar forma ciclos, esto es, a partir de un momento vuelve a repetirse la serie de números. A continuación dos formas de generación:

c1) **Cuadrado del Medio.** Tome cualquier número de cuatro dígitos y elevelos al cuadrado, del resultado quedese con las 4 cifras intermedias. El resultado vuelve elevarse al cuadrado y de nuevo tome las cuatro cifras intermedias, repita esta operación tantas veces como números aleatorios desee generar. Por ejemplo:

número	cuadrado número	nuevo
3715	13801225	8012
8012	64192144	1921
1921	3690241	9024
9024	81432576	4325
4325	18705625	7056
7056	49787136	7871
7871	61952641	9526

Debe estar claro, que en un momento, llega a generarse un número aleatorio que ya se haya generado. Lo que formaría un ciclo, que volvería a repetirse.

c2) **Métodos congruenciales o residuales.** Este método al igual que el anterior requiere de un primer número, a partir del cual se genera la serie de números aleatorios mediante la aplicación recursiva de la siguiente fórmula:

$$X_{i+1} = (\alpha X_i + \alpha) \text{ modulo } p$$

Los valores , y p dependen del sistema y equipo a emplearse para la generación de estos números aleatorios, generalmente p es el máximo valor entero que soporte el sistema. Si = 1 el método recibe el nombre de aditivo, si = 0 el método se dice multiplicativo, el valor inicial xo se le denomina núcleo.

Otra forma más sencilla aun, es utilizar el random con que cuentan los diferentes lenguajes de programación, es fácil ver que el método del cuadrado del medio, la longitud máxima del ciclo de números aleatorios es de 10,000. En el caso de los congruenciales su longitud depende del módulo elegido. Estuve una tarde generando números seudoaleatorios, probando con distintas semillas y módulos, en los dos métodos aquí descritos y en ninguno de mis intentos genere ciclos mayores a 3,000, de donde concluir que el mejor generador es el random del lenguaje.

A.3 GENERACION DE NUMEROS ALEATORIOS DE DISTRIBUCIONES FRECUENTES EN SIMULACION.

En este apartado supondremos que el problema de generar números aleatorios con distribución uniforme en el intervalo cero-uno ha sido resuelto satisfactoriamente. Se han separado en tiempos de servicio, que incluyen las distribuciones, uniforme en un intervalo, normales y exponenciales y en tiempos de llegada, donde únicamente se expone el caso de llegadas poissonianas. En todos estos casos damos una pequeña descripción y a continuación damos un algoritmo en pascal.

PARA TIEMPOS DE SERVICIO.

Uniformes en un intervalo:

1) **Caso entero:** Sea a < b, dos números enteros y deseamos generar números aleatorios enteros comprendidos entre a y b, incluyendo a ambos.

Sea yi, una serie de números aleatorios con distribución uniforme en (0,1), entonces la serie xi definida por:

$$xi = \text{mayor entero contenido en } [a + yi*(b-a+1)].$$

Tiene distribución uniforme, para los posibles valores de (a,a + 1,..,b), observe que yi, al tomar valores próximos a cero convierte al factor yi*(b-a + 1), en un valor cercano al cero, de donde xi tomará el valor de a. Por otro lado yi, no toma el valor de uno, de donde por muy cercano que sea a uno, el factor yi*(b-a + 1), sera menor que b-a + 1, de donde xi tomará el valor de b.

```
FUNCTION UNIFORME (A,B:INTEGER):INTEGER;
BEGIN
 UNIFORME: = TRUNC(A + RANDOM*(B-A + 1)
END;
```

2). **Caso continuo:** Sean a<b dos números reales cualesquiera, y queremos generar números aleatorios con distribución uniforme en ese intervalo, entonces si la serie yi tiene distribución uniforme en (0,1), la serie definida por:

$$xi = yi*(b-a);$$

tiene distribución uniforme en el intervalo (a,b).

```
FUNCTION UNIFORME_CONTINUA (A,B:REAL):REAL;
BEGIN
UNIFORME_CONTINUA:= RANDOM * (B-A)
END;
```

Normales con media M y desviación D.

Primero consideremos el caso de los normales, con media cero, y desviación uno. Si xi, es una serie de números aleatorios distribuidos uniformemente en el intervalo (0,1), entonces la serie:

$$z_i = \frac{\sum_{i=1}^{k} x_i - k/2}{(k/12)1/2}$$

tiene distribución normal (0,1). En la medida que k -- , la aproximación es mejor. Una buena aproximación es k = 12. de donde la forma anterior se reduce a:

$$z_i = \sum_{j=1}^{12} xj - 6$$

de donde si la serie yi, definida por:

$$y_i = M + z_i*D.$$

donde z_i tiene distribución normal (0,1), tiene distribución normal de media M y desviación D.

Aquí hay que tener cuidado, ya que el resultado teóricamente y mediante el algoritmo puede resultar negativo, cosa que para nuestros fines no puede ocurrir y generaria problemas, en el modelo de simulación.

```
FUNCTION NORMAL (M,D:REAL):REAL;
VAR
AUX : REAL;
I  : INTEGER;
BEGIN
 REPEAT
  AUX: = 0;
  FOR I: = 1 TO 12 DO
  AUX: = AUX + RANDOM;
  AUX: = AUX-6;
 UNTIL AUX = 0;
 NORMAL: = AUX;
END;
```

Tiempos con distribución exponenciales, de media A.

La distribución teórica está dada por $f(t) = A \exp(-At)$, de donde la acumulada de probabilidad esta dada por:

$$F(t) = 1-\exp(-At)$$

y por lo tanto la imagen inversa de $x_i \varepsilon F(t)$, esta dado por:

$$t_i = Ln(1-x_i)/(-A).$$

De donde si x_i tiene distribución uniforme en (0,1), t_i tendrá distribución exponencial de media A. Es claro que si x_i es uniformemente distribuida en (0,1), también lo será, la serie $1-x_i$. y por lo tanto la expresión anterior puede ser resumida en:

$$t_i = Ln(x_i)/(-A).$$

Observe que como xi, el logaritmo resulta negativo y al dividirlo por una cantidad negativa, obtenemos una cantidad positiva.

```
FUNCTION SERVICIO_EXPO (MEDIA : REAL):REAL;
BEGIN
 SERVICIO_EXPO:=LN(RANDOM)/(-MEDIA)
END;
```

PARA TIEMPOS DE LLEGADA.

Llegadas poissonianas (o exponenciales), con tiempo medio entre llegadas TA.

Basta observar que si TA es el tiempo medio entre llegadas, entonces A = (1/TA) es el tiempo promedio entre llegadas, siguiendo el razonamiento de la última distribución, y sustituyendo A = (1/TA) obtenemos:

$$t_i = -TA * LN(x_i)$$

```
FUNCTION LLEGADA_EXPO (TA:REAL):REAL;
BEGIN
  LLEGADA_EXPO=-TA*LN(RANDOM)
END;
```

A.4 CONCEPTOS DE TEORIA DE COLAS.

En los modelos de simulación existen una serie de situaciones en donde "alguien" o "algo" requieren de un determinado tipo de servicio para lo cual en ocasiones tienen que hacer una cola o esperar en linea ante "alguien" o "algo" que es quien proporciona dicho servicio. Es en este sentido que la teoría de colas ofrece elementos importantes de decisión en la solución a problemas tales como : disminuir o eliminar la congestión de un determinado servicio, minimizar pérdidas de tiempo debidas a operaciones deficientes, minimizar excesos de capacidad, etc.

La naturaleza de ésta situación de espera puede analizarse matemáticamente si se conocen las leyes que gobiernan las llegadas para un servicio, el orden en que son atendidos, y los tiempos para dar el servicio.

En la figura A3 se muestra la estructura cualitativa de proceso básico de lineas de espera:

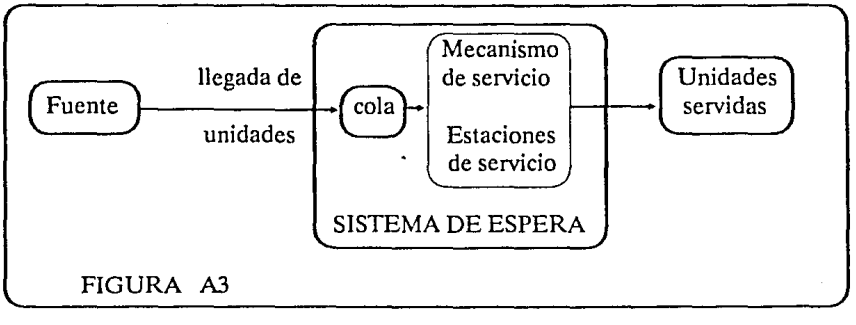


FIGURA A3

Existe una fuente de la cual provienen las unidades que requieren un determinado servicio, mismas que llegan al sistema de espera formando una cola. El mecanismo de servicio, a través de estaciones de servicio selecciona en el tiempo a una de las unidades en la cola para prestar el servicio solicitado. Una vez terminado éste, la unidad deja el sistema de espera.

La gran diversidad de situaciones en cada una de las fases del proceso descritas origina una multiplicidad de situaciones de espera, por ejemplo:

TIPOS DE ENTRADAS	NATURALEZA DEL SERVICIO	PRESTADORES DE SERVICIO
Clientes	Venta de un artículo	dependientes
Barcos	Carga y descarga	muelles
Máquina descompuesta	Reparación	mecánicos
Pacientes	Consulta	médicos

De igual forma, se pueden señalar como ejemplos de situaciones de espera los siguientes:

SERVICIO	LINEAS DE ESPERA	ESTACION DE SERVICIO
Consultorio	1	1
Peluquería	1	muchas
Gasolinera	2	1
Banco	muchas	muchas

Las características mas frecuentes del proceso de espera se mencionan a continuación:

CARACTERISTICAS DEL PROCESO DE ESPERA

1. Tipo de Llegadas. Una característica de la fuente es el tamaño del número total de unidades que solicitan servicio pudiendo ser finito o infinito. Otra característica es la forma en que las unidades llegan al sistema de espera, pudiendo distinguirse los siguientes casos:

- Llegadas de unidades al sistema a intervalos iguales de tiempo.

- Llegadas de unidades al sistema a intervalos desiguales de tiempo pero perfectamente conocidos.

- Llegadas de unidades al sistema a intervalos desiguales de tiempo cuyas probabilidades son conocidas (intervalos aleatorios).

- Llegadas de unidades al sistema a intervalos desiguales de tiempo con prob-

abilidades desconocidas y en cuyo caso no puede ser estimado.

Los casos mas típicos de distribución de llegadas con su correspondiente distribución de tiempo entre llegadas, son los siguientes:

DISTRIBUCION DE LLEGADAS	DISTRIBUCION DE TIEMPOS ENTRE LLEGADAS
Llegadas que ocurren a intervalos de tiempo iguales.	Constante
llegadas poissonianas	Exponencial
tipo poissoniano es intermedia entre las dos anteriores	Erlang (dist. gama) donde: si k = 1 exponencial si k = ∝ constante
Caso general	distribución general

En la práctica existen tipos de llegadas mas complejos: llegadas programadas a ciertos instantes de tiempo pero sujetas a variación; llegadas en grupos, colas cíclicas en las cuales un pequeño número de unidades recicla, etc.

2. Disciplina de la linea de espera. Existen diferentes disciplinas en la linea de espera, siendo el caso mas sencillo y el que normalmente se considera en modelos de espera: primero en llegar es a quien se le da primero servicio (FIFO). Otros tipos de disciplina se establecen de acuerdo con un cierto orden preferencial para otorgar servicio.

3. Mecanismo de servicio. El mecanismo de servicio puede constar de una o mas unidades de servicio, conteniendo cada unidad una o mas estaciones de servicio.

Igual que las llegadas de unidades al sistema, el tiempo de servicio (comprendido desde que la unidad entra al servicio hasta que termina) sigue la misma clasificación que en el punto 1.

CASOS MAS FRECUENTES DE DISTRIBUCION DE SALIDA:	
DISTRIBUCION DE SALIDAS TIEMPO	DISTRIBUCION DEL TIEMPO DE SERVICIO (INTERVALOS DE ENTRE SERVICIOS CONSECUTIVOS)
Salidas que ocurren a intervalos de tiempo iguales.	Constante
Salidas poissonianas	Exponencial $\mu \exp(-\mu t)$
tipo poissoniano es intermedia entre las dos anteriores	Erlang $f_k(t) = t_{k-1} (\mu k)\, k \exp(-\mu kt)/(k-1)!$ si $k = 1$ exponencial si $k = \infty$ constante
Caso general	distribución general

Otros factores que pueden influir en el servicio son:

Servicio por grupo: colas en serie en las cuales la salida de una unidad de una estación de servicio es la entrada en la siguiente estación, etc.

En general, soluciones analíticas son difíciles de obtener para problemas muy complejos debiendo usarse la simulación.

4. Notación.

Una forma de describir un proceso de espera en forma sencilla, es a través de una serie de símbolos y diagonales tales como A/B/X/Y,Z/, notación debida a Kendall (1953), en donde A indica la distribución de tiempo entre llegadas, B la distribución de probabilidades de tiempo entre servicios, X el número de estaciones de servicio paralelas, Y la restricción en la capacidad del sistema y Z la disciplina de espera.

CARACTERISTICA	SIMBOLO	EXPLICACION
Distribución de tiempo entre llegadas (A)	M	exponencial o poisson
	D	determinística
	Ek	Erlang
	G1	General independiente
Distribución de tiempo de servicio (B)	M	exponencial
	D	determinística
	Ek	Erlang
	G	General
Número de estaciones de servicio paralelas (X)	1,2..X	
Restricción en la capacidad del sistema (Y)	1,2..Y	
Disciplina de espera (Z)	FIFO	primero en entrar primero en salir
	LIFO	último en entrar primero en salir
	SIRO	servicio en forma aleatoria
	PRI	prioridad
	GD	disciplina general

Normalmente se utilizan los primeros tres símbolos, omitiéndose los símbolos Y y Z. Así M/D/2 representa un sistema de espera con tiempo entre llegadas exponencial, servicio determinístico, dos estaciones de servicio, ningún limite en la capacidad de servicio y como disciplina primero en llegar primero en servicio.

/M/M/2. Llegadas poissonianas- servicio poissoniano - 2 estaciones de servicio.

/M/G/8. Llegadas poissonianas - servicio con distribución general de tiempo entre servicios - 8 estaciones de servicio.

En la descripción de los modelos de espera se usa la siguiente nomenclatura:

E_n = Estado en el cual hay n unidades en el sistema.

λ = Relación media de llegadas = número esperado de llegadas por unidad de tiempo.

μ = Relación media de servicios = número esperado de unidades servidas por unidad de tiempo (de una estación de servicio)

$1/\lambda$ = Tiempo esperado entre llegadas.

$1/\mu$ = Tiempo esperado entre salidas - tiempo medio de servicio = tiempo esperado de servicio.

s = Número de estaciones de servicio

λ_n = Relación media de llegadas de nuevas unidades cuando hay n unidades en el sistema.

μ_n = Relación media de servicios cuando hay n unidades en el sistema.

$q_j(t)$ = Probabilidad de que existan j unidades en el sistema en el instante t.

τ = X/s Porcentaje de utilización del servicio = intensidad de tráfico = probabilidad de que el servicio esté ocupado.

BIBLIOGRAFIA

1.- COATS, P.K. "Combining an expert system with simulation to enhance planning for banking networks" Simulation, Vol.54, Num 6, junio 1990.

2.- COSS BU, R. "Simulación de Sistemas" ed. Diana 1986.

3.- COX, J. " Object Oriented Programing. An Evolutionary Approach", Addison Wesley, 1987

4.- ELDREDGE, D. MC. GREGOR, J.D. Y SUMMERS, M.K. "Applying the object- orientes paradigm to discrete event simulations using C+ + language". Simulation, Vol 54, Num 6, junio 1990.

5.- EZZELL, B. " Programming the IBM User Interface Using Turbo Pascal", Addison Wesley Publishing Company, 1989.

6.- FUENTES ZENON A. "El enfoque de sistemas en la solución de problemas", Cuadernos de planeación y sistemas, DEPFI. UNAM. 1990

7.- FUENTES Z. A. y SANCHEZ G. G. "Metodología de la Planeación Normativa", Cuadernos de planeación y sistemas, DEPFI. UNAM 1990.

8.- GORDON, G." Simulación de Sistemas", Editorial Diana, 1988.

9.- HILLIER, y LIEBERMAN. "Introducción a la Investigación de Operaciones" Ed. Mc. Graw Hill. 1989.

10.- HOFFMAN, P. y NICOLOFF, T. "Sistema Operativo MS-DOS. Guia del Usuario", Osborne Mc Graw Hill, 1984.

11.-JAMSA, K. y NAMEROFF, S. 'Turbo Pascal: Biblioteca de Programas", Mc. Graw Hill, 1986.

12.- KIVIAT, P. "Development of Discrete Digital Simulation Languages", Simulation, Vol 8, Num 2, febrero 1967.

13.- LUCAS, H. " Análisis, Diseño e Implementación de Sistemas de Información"

14.- MC. MILLAN, C. Y GONZALEZ, R.F. "Análisis de Sistemas" ed. Trillas, 1986.

15.- MEYER, B. "Object-Oriented Software Construction". Prentice-Hall International Series in Computer Science, 1988.

16.- MILLER, A. "El ABC del MS-DOS. Versión 3.3", Ventura Ediciones 1988.

17.- PAYNE, J.A. "Introduction to Simulation: Programming Techniques and Methods of Analysis" ed. Mc. Graw-Hill, 1988.

18.- P. VAN GIGCH, J. "Teoría General de Sistemas" ed Trillas 1987.

19.- SAATY, T.H. "Elementos de la Teoría de Colas". ed Aguilar, 1967.

20.- SHANNON, R.E. "Simulación de Sistemas" ed. Trillas, 1988.

21.- SMARTE, G. y REINHARDT, A. "15 Years of Bits, Bytes and Other Great Moments", Byte, vol 15, No. 9, sept. 1990.

22.- WILLIAMS, G. y WELCH, M. "A Microcomputing Timeline", Byte vol 10, No. 9, sept. 1985.