UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES

ARAGÓN

"GENERADOR DE POGRAMAS EN PASCAL"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A :

DULCE PATRICIA DOMÍNGUEZ ARIAS

ASESOR: MAT. LUIS RAMÍREZ FLORES

MÉXICO, 2007.

AGRADECIMIENTOS

A Dios principalmente, por darme la oportunidad de realizar otra de mis metas y por la familia tan maravillosa que me ha dejado formar al lado del hombre más especial del mundo y al cual sigo amando como el primer día "Mi Esposo" Ignacio Vázquez, con el que he compartido 20 años de mi vida y sin el que no habría podido terminar este proyecto. Gracias Nacho por estar conmigo, por tu amor, tu apoyo y tu confianza.

A mis adorados hijos: Jessica, Andrea y Rodrigo quienes me han hecho sentir realizada como madre y quienes son mi motivo para superarme y seguir adelante, los amo con toda el alma, son los hijos más maravillosos del mundo y espero que algún día ellos también puedan realizar todos sus sueños, poniéndole muchas ganas a todo lo que hagan, esperando guiarlos de la mejor manera posible.

A mi madre Esperanza Arias Espinosa (q.e.p.d.)a quien extraño mucho y a quien le debo lo que soy, gracias mamá donde quiera que estés.

A mi padre que aunque casi no estuvo conmigo me dio una gran lección de vida.

A mis hermanos:

Guillermo (q.e.p.d.) que fue como un segundo padre para mí.

Elizabeth quien siempre me impulsó para seguir estudiando y me dio su apoyo, sobre todo en las primeras etapas de mi educación y gracias a ella pude tener una infancia llena de ilusiones.

Guadalupe, Gerardo, Sara y Roberto, de los que aprendí mucho a lo largo de mi vida.

Inés además de mi hermana, mi amiga y consejera le doy las gracias por apoyarme siempre que lo he necesitado.

Ángel el hermano que aunque duro de carácter, siempre te tiende la mano cuando lo necesitas y del que aprendí a luchar por lo que realmente se quiere.

Antonio, a quien admiro por su gran deseo de vivir y su lucha continua para salir adelante pese a las adversidades.

A mi amiga, casi hermana Rosa Elena a quien quiero mucho y le agradezco por su apoyo y su confianza en todo tipo de situaciones y por compartir conmigo lo bueno y lo malo de la vida, espero que nuestra amistad perdure más allá de la

vida, también a Fernando su esposo y mi amigo porque sé que siempre puedo contar con ellos.

A mis compañeras de Grupo I.C.A. Mary Carmen y Lupita por su amistad incondicional.

A mi asesor y amigo Prof. Luis Ramírez Flores, que hizo que mi trabajo fuera más llevadero y del que aprendí a tratar de ser cada día mejor persona.

A mis amigos los profesores del CCH Azcapotzalco quienes también me impulsaron a terminar éste trabajo.

Y en especial a mi esposo y compañero de toda mi vida Ignacio Vázquez, sin él no hubiera podido terminar este proyecto, además de que me impulso a hacerlo, me asesoró en cada uno de los capítulos de este trabajo, a ti mi amor te doy las gracias por ser el mejor esposo y el mejor padre, te amo.

Í N D I C E

INTRODUCCIÓN

Con el paso del tiempo, el desarrollo tecnológico ha tenido un enorme auge en todos los ámbitos de nuestra sociedad, podemos apreciarlo en hospitales públicos y privados, en las grandes industrias, pequeñas empresas y por si fuera poco en el ámbito escolar, los alumnos de las escuelas en general, desde la educación básica, ya manejan el concepto de computación, visualizan bien la imagen de una computadora y además también manipulan el hardware de las mismas.

Cada vez los alumnos se empapan más de todo lo relacionado con las nuevas tecnologías e incluso la combinación de las mismas, como son la interacción de un teléfono celular con una computadora, sea del tamaño que fuere.

Los alumnos se han vuelto investigadores de lo que les gusta o les llama la atención, sin embargo lo hacen de manera mecánica y por inercia o sentido común, no razonan siempre el por qué cierto aparato realiza tal o cual función y que está detrás de ésta, tal vez un chip con algún programa especial que hace que un aparato realice determinadas funciones.

No hay algún sistema que pueda mostrarnos internamente su funcionamiento y tampoco alguien que nos ayude a descifrar los enigmas que guardan los aparatos en su interior, sobre todo los más sofisticados.

Sin embargo sabemos que podemos emplear nuestros conocimientos para poder entender el funcionamiento de las cosas y tal vez porque no, poderlo simular al mismo tiempo.

Los alumnos del Colegio de Ciencias y Humanidades, necesitan un guía para poder volverse más creativos, tal vez algo que ayude a que su capacidad de razonamiento sea más abierta y activa y menos pasiva.

Existen las herramientas para poderlo realizar, con el proyecto que se presenta en ésta tesis, llevaremos al alumno a que desarrolle mejor su razonamiento y además logramos hacer que comprenda la equivalencia entre el algoritmo de un problema y el paso del mismo a las sentencias de un lenguaje de programación, en este caso el Lenguaje de Programación Pascal.

El desarrollo del proyecto de tesis denominado “Generador de Programas en Pascal”, surgió como una inquietud de enseñar a los alumnos de la materia de Cibernética y Computación como poder elaborar un programa en pascal, a través de un sistema que pudiera pasar una línea de algoritmo o pseudocódigo al código común del lenguaje de programación.

Este sistema puede enriquecerse con más instrucciones para hacerlo más potente y también puede migrarse a otro lenguaje si así se desea después. Sin embargo resulta bastante adecuado para la realización de programas básicos y sencillos con los que se lleva al alumno a entender la programación de una manera más sencilla y amigable.

C a p í t u l o i

Reseña histórica

CAPÍTULO I. RESEÑA HISTÓRICA

PROGRAMAS DE CIBERNÉTICA Y COMPUTACIÓN I Y II

Presentación

Orientación general de los cursos

El Colegio de Ciencias y Humanidades tiene el compromiso de proporcionar a sus alumnos educación, conocimientos y habilidades que contribuyan a desenvolverse en sus actividades profesionales y personales, de tal manera que incidan en la adquisición de la cultura básica, con fundamento en los principios del modelo educativo del Colegio. Es importante que el alumno del Colegio que curse la materia de Cibernética y Computación, adquiera los conocimientos fundamentales que le permitan comprender a la cibernética como una ciencia interdisciplinaria que incluye a la computación, que ha modificado y utilizado la información en todos los campos de la actividad humana, repercutiendo en la sociedad.

El alumno comprenderá la importancia del procesamiento de la información, adquirirá conocimientos y habilidades mediante el desarrollo de estrategias que se puedan aplicar a situaciones problemáticas; comprenderá la vinculación de la matemática con la cibernética en el estudio de sistemas naturales y artificiales; adquirirá una metodología para la solución de problemas, la elaboración de algoritmos y la programación en un lenguaje de alto nivel.

Las asignaturas de Cibernética y Computación pertenecen al Área de Matemáticas, deben cursarse en dos semestres (quinto y sexto), son opcionales u obligatorias de acuerdo al área de elección. Se orientan a la síntesis de lo aprendido durante los primeros cuatro semestres y a su aplicación en el campo de la cibernética y la computación, a fomentar en los alumnos la reflexión de los procesos de aprendizaje y la construcción de conocimientos, haciendo énfasis en la investigación, el desarrollo de habilidades y conocimientos que forman parte de la cultura básica, necesarios

para el inicio de estudios superiores.

Enfoque de la materia

- Enfoque disciplinario

La materia debe propiciar en los alumnos una visión general sobre la cibernética y la computación, sus avances, perspectivas y el aprovechamiento de las herramientas computacionales en la solución de problemas.
El enfoque proporcionará al alumno: una visión global de la cibernética, mediante el estudio y análisis de sistemas naturales y artificiales, a través del diseño de modelos de sistemas; mostrando la vinculación de los circuitos lógicos, el álgebra de Boole y los sistemas de numeración en el desarrollo de la cibernética, en particular de la computación; así como, una metodología que le permita poner en práctica el análisis, el razonamiento estructurado en el desarrollo de algoritmos, la codificación en un lenguaje de programación de alto nivel, sin pretender hacer del alumno un programador.

- Enfoque didáctico

El enfoque debe orientarse para que el alumno logre los aprendizajes indicados en cada una de las unidades, haciendo énfasis en los aspectos:

- **Histórico.** Es necesario que obtengan los conocimientos del desarrollo, espacial y temporal.
- **Teórico.** Con el fin de que adquieran los conceptos necesarios para comprender los procesos de manejo de la información y los elementos de la programación.

- **Práctico**. Deben desarrollar destrezas y habilidades en la solución de problemas, así como en la elaboración y ejecución de programas.
- **Analítico**. Para evaluar los aspectos que permitan la utilización de la cibernética y la computación, sus límites y perspectivas de desarrollo tanto en la disciplina como en las diversas ramas del conocimiento; así como los

procesos de solución, los programas y sus resultados.

Es necesario que durante todo el curso, se consideren los principios del Colegio: aprender a aprender, aprender a hacer y aprender a ser. Se sugiere realizar las estrategias de aprendizajes especificadas en las unidades de los programas.

Contribución de la materia al perfil del egresado

En el proceso educativo, la enseñanza de las matemáticas contribuye a la formación de la personalidad del adolescente mediante el desarrollo de conocimientos, habilidades y destrezas intelectuales, la evolución de sus formas de pensamiento y la adquisición de valores, actitudes y normas. En particular, la materia de Cibernética y Computación propiciará:

- La valoración de la dimensión tecnológica de los conocimientos que adquiere y aplicación de los mismos en la solución de problemas.
- La valoración del conocimiento científico.
- La aplicación de los conocimientos en distintos ámbitos de su actividad, con actitudes de seguridad en si mismo y de autoestima.
- La comprensión de las relaciones entre distintos campos del saber, el proceso de evolución histórica de los conocimientos y la relación con la sociedad en la cual se producen.
- La habilidad de resolver problemas y establecer relaciones con conocimientos adquiridos, planteando métodos de solución y su comprobación a través de procedimientos adecuados.
- El aprender por si mismo, adquirir habilidades de trabajo intelectual y conocimientos específicos que le permitan aumentar o construir otros y generar estrategias propias para alcanzar aprendizajes cada vez mas independientes y complejos.
- Fundamentar con racionalidad, responsabilidad y rigor sus conocimientos e ideas.

- El asimilar en su manera de ser, hacer y pensar, los conocimientos y habilidades que lo lleven a mejorar su propia interpretación del mundo y a adquirir madurez intelectual.
- Desarrollar un pensamiento lógico, reflexivo, crítico y flexible que se manifiesta en su capacidad para innovar en las diversas esferas de su actividad.
- Utiliza adecuadamente los algoritmos, de tal forma que resuelve los problemas y expresa sus resultados y conclusiones de manera adecuada.
- La habilidad para el manejo de estrategias de solución de problemas usando la computadora.
- El interés por la lectura y comprensión de textos diversos, particularmente científicos y de divulgación.

[1]

[1] *Programas de estudio de Cibernética y Computación I y II*. UNAM. 2006, P.p. 1-14

I.2.- DEFINICIÓN DE PROBLEMA, TIPOS DE PROBLEMAS Y HERRAMIENTAS COMPUTACIONALES Y NO COMPUTACIONALES QUE RESUELVEN PROBLEMAS DE DIVERSOS TIPOS.

Un **problema** suele ser un asunto del que se espera una solución. Puede referirse a:

- en matemática, un **problema** es una pregunta sobre objetos y estructuras matemáticas que requiere una explicación y demostración. Estas preguntas pueden ser muy específicas ("¿Cuáles son las soluciones reales de $x^2 + 1 = 0$?") o bastante generales ("¿Por qué estos números aparecen en situaciones aparentemente muy distintas? Formule y demuestre una conjetura").
- en la sociedad, un **problema** puede ser algún asunto social particular que, de ser solucionado, daría lugar a beneficios sociales como una mayor productividad o una menor confrontación entre las partes afectadas. Para exponer un problema, y hacer las primeras propuestas para solucionarlo, se debe escuchar al interlocutor para obtener más información, y hacer preguntas, aclarando así cualquier duda.
- en religión, un **problema** puede una aparente contradicción entre dos dogmas, como ocurre en el problema del mal (un dios omnibenevolente, omnisciente y todopoderoso que permite la existencia de maldad y sufrimiento) y el problema del infierno (el mismo dios, que permite que algunos sean torturados eternamente en el infierno).
- Conjunto de hechos o circunstancias que dificultan la consecución de algún fin.
- En filosofía se refiere a algo inquietante que perturba la paz de quien lo tiene.

En un problema de Programación Lineal, según sean las restricciones, se obtendrán poliedros diferentes, acotados o no, y según sea la posición de la función objetivo respecto de dicho poliedro se pueden originar diferentes situaciones. Según el tipo de soluciones que presenten un problema de Programación Lineal puede ser:

Factible: si existe la región factible. En este caso nos podemos encontrar:

Óptimo finito y único. La solución óptima está formada por un único punto con coordenadas reales.

Múltiples óptimos. Un problema de Programación Lineal puede tener más de un óptimo. Además, o bien el problema tiene un único óptimo, o bien, tiene infinitos óptimos.

Óptimo infinito. Un problema de Programación Lineal puede tener un óptimo no finito, es decir, la función objetivo puede tomar, un valor tan grande o tan pequeño como se quiera sin abandonar la región factible.

Región factible no acotada, óptimo finito. La no acotación de la región factible no implica necesariamente óptimo infinito. Puede ocurrir que la función objetivo alcance el óptimo en la zona acotada de la región factible.

Región factible no acotada, óptimo finito e infinito. Puede darse el caso que todos los puntos de una de las semirrectas que determinan la región factible no acotada sean solución del problema.

No factible. Región factible vacía. El conjunto de restricciones de un problema de Programación Lineal puede ser incompatible, conduciendo a una región factible vacía.

PÁGINA ANTERIOR	ÍNDICE	PÁGINA SIGUIENTE

1.INTRODUCCIÓN	2.PLANTEAMIENTO DIDÁCTICO	3. UN POCO DE HISTORIA	4. DEFINICIÓN Y TERMINOLOGÍA	5. TIPOS DE PROBLEMAS	6. INECUACIÓN
7.SISTEMAS DE INECUACIONES	8. MÉTODOS DE SOLUCIÓN	9.APLICACIONES	10. EL ALGORITMO DEL SIMPLEX	11.EJERCICIOS	12. BIBLIOGRAFÍA

HERRAMIENTAS BÁSICAS PARA LA SOLUCIÓN DE PROBLEMAS
Contenido:

INTRODUCCIÓN

La evolución del concepto de calidad aplicado a la industria, y ahora a los servicios, muestra claramente que se ha pasado de una etapa, en donde la calidad era aplicada totalmente al control realizado al final de las líneas de producción, a otra donde aplicamos calidad total a todo dentro de la organización. Por ende, ya se habla de calidad de vida en el trabajo, calidad de vida en los servicios y calidad ambiental.

Recordemos que el concepto de calidad hoy en día, es aplicado en el ámbito industrial, como el logro de hacer las cosas bien la primera vez. Y se aplica control de calidad sobre las operaciones desde el diseño. Hasta que se obtiene el producto final e inclusive se habla de la calidad en la atención al cliente.

El camino que nos lleva hacia la Calidad Total crea una nueva cultura, establece y mantiene un liderazgo, desarrolla al personal y lo hace trabajar en equipo, además de enfocar los esfuerzos de calidad total hacia el cliente y a planificar cada uno de los pasos para lograr la excelencia en sus operaciones.

El hacer esto exige vencer obstáculos que se irán presentando a lo largo del camino. Estos obstáculos traducidos en problemas se deben resolver conforme se presentan evitando con esto las variaciones del proceso. Para esto es necesario basarse en hechos y no dejarse guiar solamente por el sentido común, la experiencia o la audacia. Basarse en estos tres elementos puede ocasionar que al momento de obtener un resultado contrario al esperado nadie quiera asumir responsabilidades.

De allí la importancia de basarse en hechos reales y objetivos, además de que surge la necesidad de aplicar herramientas de solución de problemas adecuadas y de fácil comprensión.

Las herramientas y técnicas cualitativas y no cuantitativas son las siguientes:

1. Recolección de datos.
2. Lluvia/Tormenta de ideas (Brainstorming).
3. Diagrama de Paretto.
4. Diagrama de Ishikawa.
5. Diagrama de flujo.
6. Matriz de relación.
7. Diagrama de comportamiento
8. Diagrama de Gantt.
9. Entrevistas.
10. Listas checables.
11. [2]Presentación de resultados.

La experiencia de los especialistas en la aplicación de estas herramientas señala que bien utilizadas y aplicadas, con la firme idea de estandarizar la

[2] *www.computacion*

solución de problemas, los equipos pueden ser capaces de resolver hasta el 95% de los problemas.

RECOLECCIÓN DE DATOS

CONCEPTO

Es una recolección de datos para reunir y clasificar las informaciones según determinadas categorías de un evento o problema que se desee estudiar. Es importante recalcar que este instrumento se utiliza tanto para la identificación y análisis de problemas como de causas.

USO

Hace fácil la recopilación de datos y su realización de forma que puedan ser usadas fácilmente y ser analizadas automáticamente. Una vez establecido el fenómeno que se requiere estudiar e identificadas las categorías que lo caracterizan, se registran los datos en una hoja indicando sus principales características observables.

Una vez que se ha fijado las razones para recopilar los datos, es importante que se analice las siguientes cuestiones:

- La información es cuantitativa o cualitativa.
- Cómo se recogerán los datos y en que tipo de documentos se hará.
- Cómo se utilizará la información recopilada.
- Cómo se analizará.
- Quién se encargará de recoger los datos.
- Con qué frecuencia se va a analizar.
- Dónde se va a efectuar.

OTROS NOMBRES

- Hoja de recogida de datos
- Hoja de registro
- Verificación
- Chequeo o Cotejo

PROCEDIMIENTO

1. Identificar el elemento de seguimiento
2. Definir el alcance de los datos a recoger.
3. Fijar la periodicidad de los datos a recolectar.
4. Diseñar el formato de la hoja de recogida de datos, de acuerdo a la cantidad de información a escoger, dejando espacio para totalizar los datos, que permita conocer: las fechas de inicio y termino, las probables interrupciones, las personas que recoge la información, la fuente etc.

LLUVIA DE IDEAS

CONCEPTO

Técnica que consiste en dar oportunidad, a todos los miembros de un grupo reunido, de opinar o sugerir sobre un determinado asunto que se estudia, ya sea un problema, un plan de mejoramiento u otra cosa, y así se aprovecha la capacidad creativa de los participantes.

USO

Se pueden tener dos situaciones ante la solución de un problema:

1. Que la solución sea tan evidente que sólo tengamos que dar los pasos necesarios para implementarla, y
2. Que no tengamos idea de cuáles pueden ser las causas, ni las soluciones.

Es aquí donde la sesión de tormenta de ideas es de gran utilidad. Cuando se requiere preseleccionar las mejores ideas.

OTROS NOMBRES

- Brain Storming
- Tormenta de ideas

PROCEDIMIENTO

1. Nombrar a un moderador del ejercicio.
2. Cada miembro del equipo tiene derecho a emitir una sola idea por cada turno de emisión de ideas.
3. No se deben repetir las ideas.
4. No se critican las ideas.
5. El ejercicio termina cuando ya no existan nuevas ideas.
6. Terminada la recepción de las ideas, se les agrupa y preselecciona conforma a los criterios que predefina el equipo.

DIAGRAMA DE PARETTO

CONCEPTO

Gráfico cuyas barras verticales están ordenadas de mayor a menor importancia, estas barras representan datos específicos correspondientes a un problema determinado, la barra más alta esta del lado izquierdo y la más pequeña, según va disminuyendo de tamaño, se encuentra hacia la derecha.

USO

Ayuda a dirigir mayor atención y esfuerzo a problemas realmente importantes, o bien determina las principales causas que contribuyen a un problema determinado y así convertir las cosas difíciles en sencillas. Este principio es aplicable en cualquier campo, en la investigación y eliminación de causas de un problema, organización de tiempo, de tareas, visualización del antes y después de resuelto un problema, o en todos los casos en que el efecto final sea el resultado de la contribución de varias causas o factores.

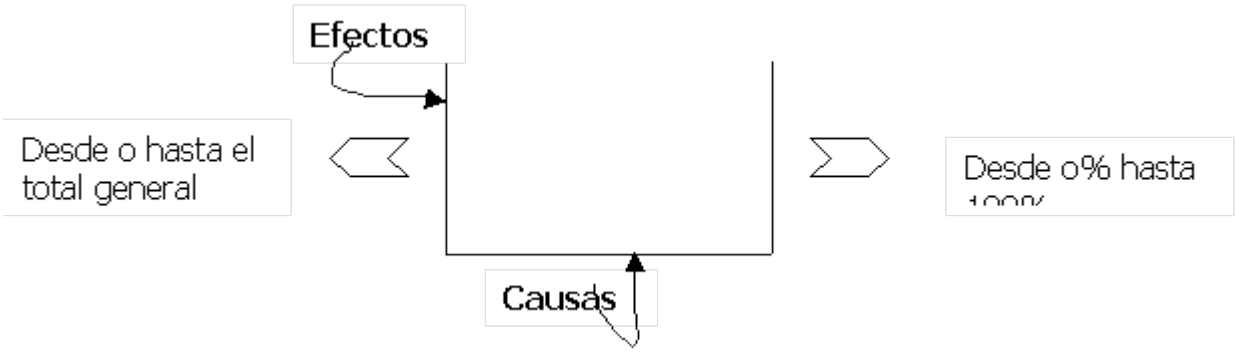
PROCEDIMIENTO

1. Decidir qué problemas se van a investigar y cómo recoger los datos.
2. Diseñar una tabla de conteo de datos (totales).
3. Elaborar una tabla de datos.

Tipo de Reclamo	Número	Número Acumulado	%	% Acumulado
B	8	8	28,57	28,57
C	7	15	25.00	53.57
D	6	21	21.43	75.00
A	4	25	14.29	89.29
E	3	28	10.71	100.00

- Lista de ítems
- Totales individuales
- Totales acumulados
- Composición porcentual
- Porcentajes acumulados

4. Organizar los ítems de mayor a menor.
5. Dibujar dos ejes verticales y uno horizontal



6. Construir un diagrama de barras.
7. Dibujar la curva acumulada (curva de Pareto).
8. Escribir cualquier información necesaria.

DIAGRAMA DE ISHIKAWA

CONCEPTO

Técnica de análisis de causa y efectos para la solución de problemas, relaciona un efecto con las posibles causas que lo provocan.

USO

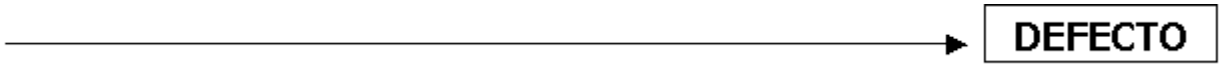
Se utiliza para cuando se necesite encontrar las causas raíces de un problema. Simplifica enormemente el análisis y mejora la solución de cada problema, ayuda a visualizarlos mejor y a hacerlos más entendibles, toda vez que agrupa el problema, o situación a analizar y las causas y subcausas que contribuyen a este problema o situación.

OTROS NOMBRES

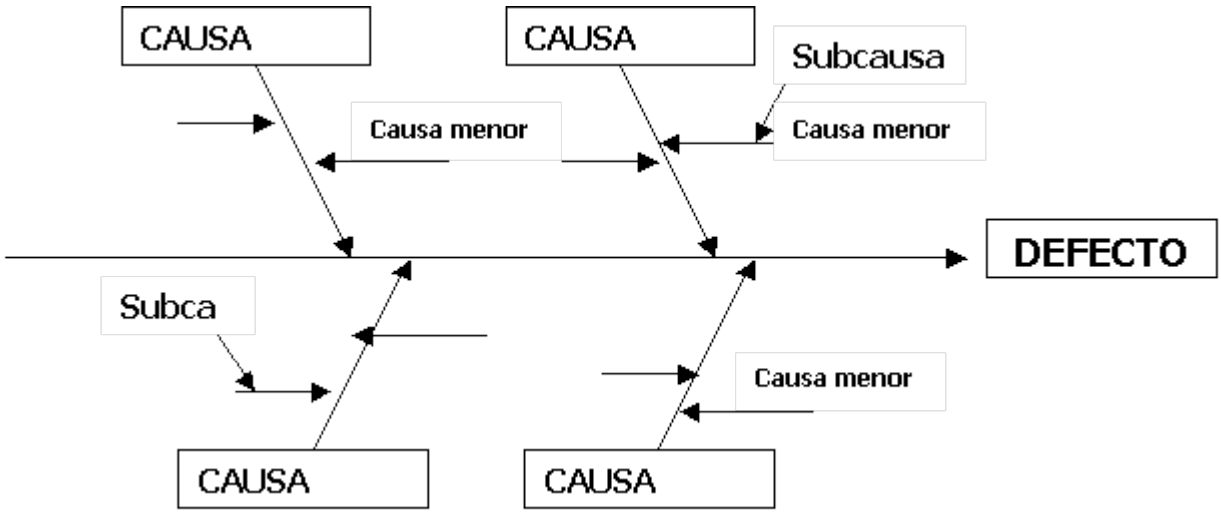
- Diagrama de espina de pescado
- Diagrama Causa Efecto

PROCEDIMIENTO

1. Ponerse de acuerdo en la definición del efecto o problema
2. Trazar una flecha y escribir el “efecto” del lado derecho



3. Identificar las causas principales a través de flechas secundarias que terminan en la flecha principal
4. Identificar las causas secundarias a través de flechas que terminan en las flechas secundarias, así como las causas terciarias que afectan a las secundarias



5. Asignar la importancia de cada factor
6. Definir los principales conjuntos de probables causas: materiales, equipos, métodos de trabajo, mano de obra, medio ambiente (4 M`s)
7. Marcar los factores importantes que tienen incidencia significativa sobre el problema
8. Registrar cualquier información que pueda ser de utilidad

MATRIZ DE RELACIÓN

CONCEPTO

Gráfico de filas y columnas que permite priorizar alternativas de solución, en función de la ponderación de criterios que afectan a dichas alternativas.

USO

· Cuando se requiere tomar decisiones más objetivas.
· Cuando se requiere tomar decisiones con base a criterios múltiples.

OTROS NOMBRES

· Matriz de priorización
· Matriz de selección

PROCEDIMIENTO

1. Definir las alternativas que van a ser jerarquizadas
2. Definir los criterios de evaluación
3. Definir el peso de cada uno de los criterios
4. Construir la matriz

	CRITERIOS				
SOLUCIONES	1	4	2	3	TOTAL
Envío de solicitud por	3	2	1	1	
Envío de solicitud vía Faz o E -	3	2	3	2	
Envío de solicitud vía correo	3	1	3	1	

5. Definir la escala de cada criterio
6. Valorar cada alternativa con cada criterio (usando la escala definida anteriormente)
7. Multiplicar el valor obtenido en el lado izquierdo de las casillas, por el peso de cada criterio y anotarlo a la derecha de cada casilla
8. Sumar todas las casillas del lado derecho y anotar el resultado en la casilla Total
9. Ordenar las alternativas de mayor a menor

DIAGRAMA DE COMPORTAMIENTO

CONCEPTO

Herramienta que permite graficar los puntos del comportamiento de una variable, de acuerdo a como se van obteniendo.

USO

· Para representar visualmente el comportamiento de una variable
· Evaluar el cambio de una proceso en un período

NOMBRES

· Diagrama de Tendencias

PROCEDIMIENTO

1. Decidir qué problema se va a monitorear y cómo se van a recoger los datos
2. Mantener el orden de los datos, tal como fueron recolectados
3. Dibujar un eje vertical y uno horizontal (Eje X Tiempo - Eje Y Medida)
4. Marcar los puntos. Un punto marcado indica ya sea la medición o cantidad observada en un tiempo determinado
5. Unir las líneas de puntos
6. Escribir en el diagrama cualquier información necesaria

DIAGRAMA DE GANTT

CONCEPTO

Gráfico que establece el orden y el lapso en que deben ejecutarse las acciones que constituyen un proyecto.

USO

· Permite vigilar el cumplimiento de un proyecto en el tiempo.
· Permite determinar el avance en un momento dado.

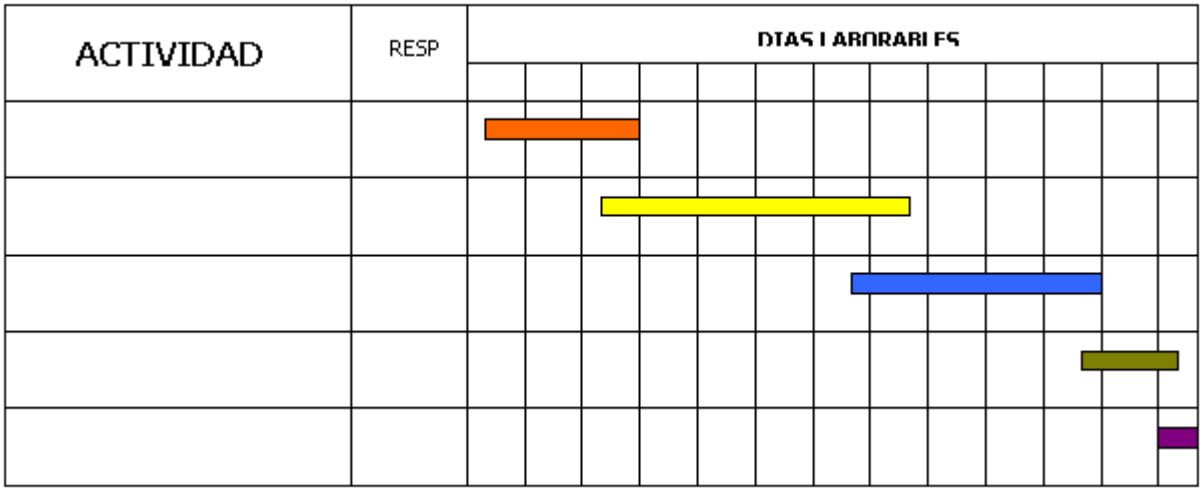
OTROS NOMBRES

· Cronograma de actividades

PROCEDIMIENTO

1. Identificar y listar todas las acciones que se deben realizar para cumplir con un proyecto
2. Determinar la secuencia de ejecución de las acciones
3. Definir los responsables de ejecutar cada acción
4. Escoger la unidad de tiempo adecuada para trazar el diagrama
5. Estimar el tiempo que se requiere para ejecutar cada acción

6.



Trasladar la información anterior a las ubicaciones correspondientes en el diagrama

ENTREVISTAS

CONCEPTO

Técnica que permite reunir información directamente con el involucrado en el proceso.

USO

Obtener información de clientes o proveedores de un proceso.

PROCEDIMIENTO

1. Planear la entrevista. Determinar que información se necesita recopilar.
2. Elaborar una guía para la entrevista (introducción, preguntas relacionadas con el tema). Elaborar una prueba piloto.
3. Seleccionar las personas que más conozcan sobre el tema.
4. Programar la entrevista. Planear el tiempo necesario para realizar la entrevista.
5. Ubicar un lugar apropiado para realizar la entrevista sin interrupciones.
6. Invitar al entrevistado, informarle del objetivo, fecha y lugar donde se realizará la entrevista.
7. Realizar la entrevista (sea puntual, cordial y desarrolle la guía para la entrevista, luego resuma y permítale al entrevistado hacer comentarios. Dele las gracias.)

LISTAS CHECABLES

CONCEPTO

Método, lista u hoja de información para lograr que nada se nos olvide ni se omita, en la cual la información consignada es de fácil análisis y verificación. Las podemos encontrar con diferencias sencillas y de tres tipos:

- Guías para la realización secuencial de operaciones, observaciones o verificaciones.
- Tablas o formatos para facilitar la recolección de los datos.
- Dibujos o esquemas para señalar la localización de puntos de interés.

USO

- Muestra una secuencia sistemática de hacer las cosas.
- Facilita la recolección de datos.
- Relaciona pasos o elementos que constituyen el todo de un proyecto o de una preparación.
- Proporciona un medio de seguimiento y control del avance de un proyecto.

Nº	Oficina	Listo	Por remodelar
1	Salón de conferencias		
2	Dirección Nal. Ejecutiva de Desarrollo		
3	Consultores de la AID		
4	Despacho del Contralor		
5	Departamento de Planillas		
6	Pagos		

I.3 Programación Estructurada, Diagramas de flujo y Pseudocódigo.

Programación Estructurada. Diagramas de Flujo y Pseudocódigo

Programación Estructurada

Diagramas de Flujo

- Es la representación gráfica de los pasos que deben seguirse para resolver un problema.
- El traducir una descripción narrada a diagrama de flujo agrega claridad y precisión a la descripción de una tarea.
- Además, al elaborar el diagrama de flujo, se descubren situaciones que no habían sido consideradas.

Programación Estructurada

Diagramas de Flujo

- En la elaboración de éstos, la simbología juega un papel muy importante, ya que debe estar adecuada a ciertos estándares, con el fin de que sea entendida por cualquier persona dedicada al campo de la computación.
- En los diagramas de flujo se utilizan figuras geométricas conectadas por líneas.
- Cada una de las figuras representa una etapa en la solución del problema; dentro de ellas se anotan indicaciones. Las líneas señalan el flujo de la información.

Programación Estructurada

Diagramas de Flujo

- En la actualidad se emplean poco, pero resultan muy útiles cuando se comienza en el estudio de la programación.
- El problema con los diagramas de flujo es que a medida que crece la complejidad de las proposiciones, también crece el detalle con el que hay que dibujarlas.
- Esto llegaba a convertirlos en figuras fraccionadas (pues de otro modo no cabrían en la hoja), y difíciles de seguir y entender.

Programación Estructurada

Diagramas de Flujo

- El equivalente simbólico de la flecha se llama goto (ir a) y está, en términos generales, excluido de la programación moderna debido a su poder "desestructurante" y caótico sobre los programas (aunque si se usa con cuidado puede llegar a ser de alguna utilidad).
- Los símbolos utilizados han sido normalizados por el Instituto Norteamericano de Normalización (ANSI) y los más frecuentes utilizados son:

Programación Estructurada

Diagramas de Flujo

INICIO O FIN DE PROCESO:
Indica el inicio o el fin de un Diagrama de Flujo. Dentro de la figura se debe escribir "inicio" o fin; según sea el caso.

ACCIONES U OPERACIONES:
Se utilizan para señalar las actividades, los pasos o las instrucciones en forma secuencial

ENTRADA / SALIDA DE INFORMACION:
Representa la entrada y salida de datos en la computadora.

Programación Estructurada

Diagramas de Flujo

LINEAS DE FLUJO:
Indican el sentido o dirección que lleva el diagrama de flujo desde su inicio hasta su fin.

DECISION:
Permite decidir entre 2 opciones o caminos a seguir.

CONECTOR :
Indica la continuidad del Diagrama De Flujo en una misma página. Dentro de la circunferencia se anota un número o una letra.

Programación Estructurada

Diagramas de Flujo

Salida (impresión):
Indica un resultado mostrado como consecuencia del proceso llevado a cabo.

CICLO REPETITIVO (for):
Indica la utilización de una estructura repetitiva

CONECTOR DE PAGINA:
Indica la continuación del diagrama de flujo de una página a otra. Se debe especificar con letra o número esta secuencia.

Programación Estructurada

Diagramas de Flujo

Reglas Básicas para la construcción de DF

- Todo diagrama debe tener un inicio y un fin
- Las líneas de conexión o de flujo deben ser siempre rectas, si es posible verticales y horizontales nunca cruzadas o inclinadas; para conseguir lo anterior es necesario apoyarse en conectores.
- Las líneas que enlazan los símbolos entre sí deben estar todas conectadas.

Programación Estructurada

Diagramas de Flujo

- Se deben dibujar todos los símbolos de modo que se pueda seguir el proceso visualmente de arriba hacia abajo (diseño de top-dow) y de izquierda a derecha.
- Realizar un gráfico claro y equilibrado
- Evitar la terminología de un lenguaje de programación o máquina.

Programación Estructurada

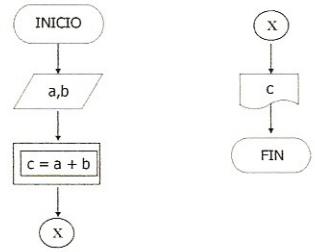
Diagramas de Flujo

- Utilizar comentarios al margen (si es necesario) para que éste sea entendible por cualquier persona que lo consulte.
- A cada bloque o símbolo se accede por arriba y/o por la izquierda y se sale por abajo y/o por la derecha.
- Si el diagrama abarca más de una hoja es conveniente enumerarlo e identificar de donde viene y a donde se dirige.

Programación Estructurada

Diagramas de Flujo

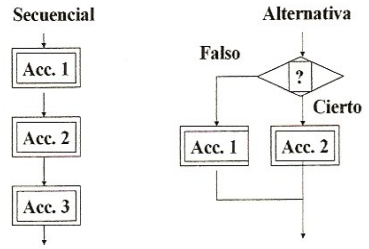
- **Ejemplo de un Diagrama de Flujo:**
  - **PROBLEMA:** Elaborar un programa que calcule la sumatoria de 2 números:



Programación Estructurada

Diagramas de Flujo

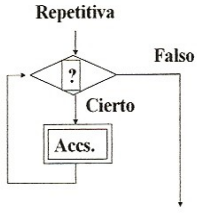
Estructuras de control en Diagramas de Flujo



Programación Estructurada

Diagramas de Flujo

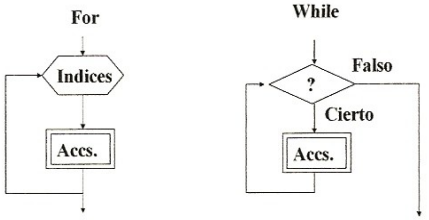
Estructuras de control en Diagramas de Flujo



Programación Estructurada

Diagramas de Flujo

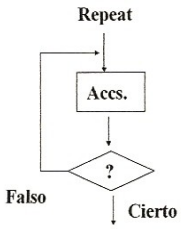
Estructuras Repetitivas en Diagramas de Flujo



Programación Estructurada

Diagramas de Flujo

Estructuras Repetitivas en Diagramas de Flujo



Programación Estructurada

Pseudocódigo

- Los algoritmos se deben describir en un lenguaje que se parezca más al lenguaje utilizado para escribir programas de computador.
- Es decir, un *lenguaje de pseudoprogramación*, una imitación del código de las computadoras al cual se le conoce como **pseudocódigo.**

Programación Estructurada

Pseudocódigo

- El **pseudocódigo** se concibió para superar las dos principales desventajas del diagrama de flujo:
  - *el diagrama de flujo es lento de crear y*
  - *difícil de modificar sin un nuevo redibujo.*
- Por otra parte el pesudocódigo es más fácil de utilizar ya que es similar al español -o al inglés, catalán, alemán o frances, dependiendo del caso.

Programación Estructurada

Pseudocódigo

- Al contrario que los lenguajes de programación de alto nivel, como pascal o Basic, no existe un conjunto de reglas que definan con precisión lo que es y lo que no es pseudocódigo.
- Varía de un programador a otro y de que tan próxima sea la descripción al lenguaje de programación.
- El pseudocódigo es una mezcla de lenguaje natural y símbolos, términos y otras caracteristicas comunmente utilizadas en uno o más lenguajes de alto nivel.

Programación Estructurada

Pseudocódigo

- Tipicamente se encuentran las características en diferentes pseudocódigos que se pueden encontrar en libros de texto de programación.
- El pseudocódigo requiere de ciertos símbolos privilegiados que ya tienen significado preciso y establecido de antemano.
- A tales indicadores del pseudocódigo se les conoce como "**palabras clave**" (keywords).

Programación Estructurada

Pseudocódigo

- Es necesario que exista una palabra clave para la selección y otra para la iteración condicional, así como para las instrucciones adicionales y otras estructuras de control.
- Por ejemplo, la palabra "**escribe**" es una palabra clave que ya tiene significado predefinido, a diferencia de la palabra ALFA, que es una variable libre.

Programación Estructurada

Pseudocódigo

- Se pretenderá uniformizar el pseudocódigo utilizando la siguiente simbología :

Intrucción en Inglés	Pseudocódigo en español
Begin	Inicio
End	Fin
Read / Input	Leer / Entrada de Datos
Write / Print	Escribir / Salida de Datos
If _____ then	Si _____ entonces
Else	Sino / Caso Contrario
For	Desde
While	Mientras
Repeat	Repetir
Until _____	Hasta_____

Programación Estructurada

Pseudocódigo

- El algoritmo comienza con la palabra **inicio** y termina con la palabra **fin**. Entre estas palabras, se escribe una instrucción (acción) por línea o se separan con un punto y coma.
- La Línea encerrada entre llaves ({...}) se denomina *comentario* : es una información al lector del programa y no realiza ninguna instrucción ejecutable, sólo tiene efectos de documentación interna del programa.
- La asignación se llevara a cabo mediante el signo = EJEMPLO: A = 10, a la variable A se le asigna el valor de 10.

Programación Estructurada

Pseudocódigo

- Por lo tanto, el **Pseudocódigo** a utilizar incluira:
  - Nombre del programa
  - Sección de declaraciones (variables y constantes)
  - Algoritmo

Programación Estructurada

Pseudocódigo

- **Ejemplo en Pseudocógico:**
- **PROBLEMA:** Elaborar un programa que calcule la sumatoria de 2 números:

```
Programa Suma dos numeros
Variables N1, N2,  S enteros
Inicio
    Leer  N1
    Leer  N2
    S = N1+ N2
    Escribir S
Fin
```

- Donde :
- N1 = Variable que recibe el primer número
- N2 = Variable que recibe el segundo número
- A la variable S se le asigna la suma de los dos números

Programación Estructurada

Ejercicios

- Se desea obtener una Tabla con las depreciaciones acumuladas y los valores reales de cada año de un automóvil comprado en $1.800.000 pesos en el año 1992, durante los seis años siguientes; suponiendo un valor de recuperación de $120.000. Realizar el análisis del problema, conociendo la fórmula de la depreciación anual constante D para cada año de vida útil.

$$D = \frac{\text{costo} - \text{valor de recuperación}}{\text{vida útil}}$$

Programación Estructurada

Ejercicios

Año	Depreciación	Depreciación Acumulada	Valor anual
1 (1992)	280.000	280.000	1.520.000
2 (1993)	280.000	560.000	1.240.000
3 (1994)	280.000	840.000	960.000
4 (1995)	280.000	1.120.000	680.000
5 (1996)	280.000	1.400.000	400.000
6 (1997)	280.000	1.680.000	120.000

Programación Estructurada

Ejercicios

Entradas

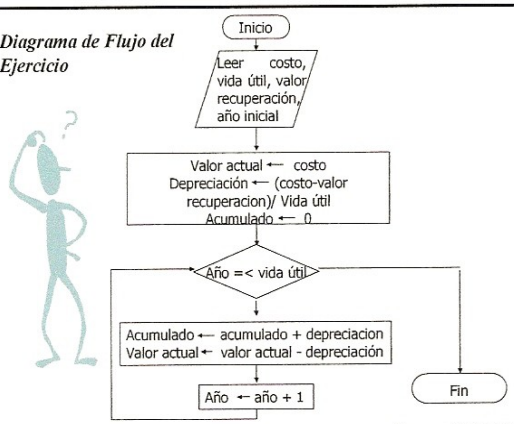
- Costo original
- Vida útil
- Valor de Recuperación

  - Procesos
  - Depreciación en cada año
  - Cálculo de la depreciación acumulada
  - Cálculo del valor del automovil en cada año

    - Salidas
    - Depreciación anual por año
    - Depreciación acumulada en cada año
    - Valor del automóvil en cada año

Programación Estructurada

Diagrama de Flujo del Ejercicio



Programación Estructurada

Pseudocódigo del Ejercicio

```
Calculo de Depreciacion
Introducir Costo
          Vida útil
          Valor de Recuperacion
Imprimir Cabeceras de tabla
Establecer el valor incial del Año
Calcular Depreciación
Mientras valor año =< vida util hacer
   Calcular depreciacion acumulada
   Calcular valor actual
   Imprimir una linea de la tabla con los valores calculados
   Incrementar el valor del año en uno
Fin de mientras
```

Programación Estructurada

Ejercicios

- Realice los siguientes ejercicios en Diagrama de Flujo y Pseudocódigo:
- Calcular el exponencial de un número (a^b), considerando todos los casos posibles:
  - Ingreso de números negativos
  - Ingreso de valores igual a 0 (a=0, b=0)
- Calcule e imprima los primeros n números primos, considere que n es ingresado por el usuario.

Programación Estructurada

Ejercicios

- Se pide determinar el mayor de tres números ingresados por el usuario. Considere que pueden ser iguales
- Calcule el factorial de un número ingresado por el usuario.

Programación Estructurada

Capítulo ii

Análisis

CAPÍTULO II.- ANÁLISIS.

II.1.- SITUACIÓN QUE PRESENTA EL ESTUDIANTE PARA COMPRENDER EL CONCEPTO DE PROBLEMA.

Es difícil hacer que los alumnos comprendan al 100% uno de los conceptos principales en programación: El concepto de variable y además su contenido, la definición de los nombres de variables y el significado de cada una.

A pesar de que el concepto de variable lo manejan desde la secundaria y es retomado en el bachillerato y además lo utilizan en materias como Matemáticas desde la I hasta la VI, en su modalidad de variables dependientes e independientes, no logran asimilar bien dicho concepto.

Ahora bien, en las materias de Cibernética y Computación I y II se requieren este tipo de definiciones, las cuales se repasan continuamente en clase, sin embargo y a pesar de ser repetitivo, no se logra que los alumnos comprendan a la perfección el concepto de variable, pero por lo menos lo asimilan mejor.

En la materia de Cibernética II, generalmente se trabaja con lenguaje de programación Pascal, a los alumnos se les enseñan los conceptos básicos para utilizar el lenguaje, así como las instrucciones y el formato de las mismas, un ejemplo y muchos ejercicios prácticos para que comprenda mejor la utilización de Pascal. Aunque es un lenguaje sencillo, a la mayoría de los estudiantes les es difícil elaborar el programa, realizan el algoritmo, el diagrama de flujo, el pseudocódigo, pero no así el programa, por ello la herramienta que se ha creado “GENERADOR DE PROGRAMAS EN PASCAL” les ayuda a la realización del programa a través de su algoritmo y además los lleva paso a paso a generarlo, convirtiendo sus sentencias en instrucciones de lenguaje de programación y también con el menú de ayuda, ubicar al estudiante sobre la manera de utilizar la instrucción.

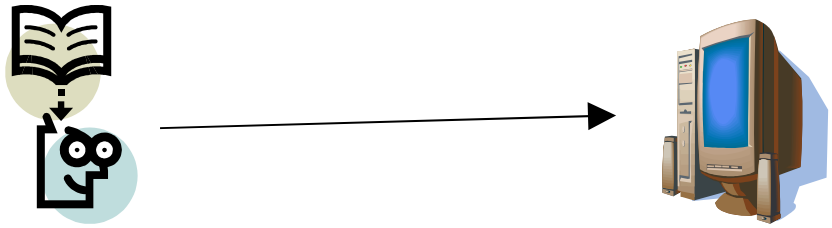
Cuando los alumnos comienzan a trabajar con variables, en cualquiera de sus materias, a veces les resulta complicada su definición dentro de un problema, sin embargo, al final del primer semestre de la materia de Cibernética y Computación I, logran asimilar mejor el concepto de variable, su contenido y el nombre que se defina para cada una de ellas, el número de variables que deben utilizar en su algoritmo y sobre todo la sintáxis de las mismas.

Para que el alumno pueda utilizar mejor el “GENERADOR DE PROGRAMAS EN PASCAL”, deberá realizar su algoritmo, con sus respectivas variables declaradas y las fórmulas principales a utilizar, ya que el “GENERADOR DE PROGRAMAS EN PASCAL” les servirá exclusivamente para elaborar programas con código sencillo, con instrucciones básicas y con ciclos simples como el FOR.

Como ya se ha mencionado a lo largo de este proyecto, uno de los problemas que enfrentan los alumnos de Cibernética y Computación II es la programación, por lo que este proyecto pretende facilitar la tarea del alumno para aprender a programar, en este caso en el lenguaje de programación Turbo Pascal.

Si el sistema logra cumplir con las perspectivas y objetivos del curso de programación, entonces el alumno podrá aprender a programar en Pascal creando su propio código.

Además, si se logra tener la atención completa del alumno, este podrá incluso con esta herramienta, programar en cualquier otro lenguaje de programación.



Alumno Software(Generador) Lenguaje de programación Computadora

II.2 ANÁLISIS COSTO – BENEFICIO

Para la Universidad Nacional Autónoma de México, y en especial al Colegio de Ciencias y Humanidades Plantel Azcapotzalco, la implantación del sistema no genera costo alguno, ya que el Colegio cuenta con la infraestructura necesaria para que el sistema desarrollado trabaje adecuadamente en un espacio y un equipo que ya están predeterminados única y exclusivamente al área de cómputo.

Si hablamos de beneficios, podemos decir entonces que son muchos, principalmente que no produce costos, que los profesores pueden utilizar el software y hacer que el alumno se interese más en la clase, debido a que le será más amena y provechosa, se evitará la deserción, puesto que a los estudiantes les servirá como un espacio de aprender en forma de juego, el índice de reprobación se verá disminuido y sobre todo egresarán del colegio alumnos mejor preparados para cualquier carrera universitaria que contenga materia de programación, no importando el lenguaje que se maneje.
Además los alumnos que elijan carreras distintas a las mencionadas, también podrán obtener provecho de lo aprendido con el Generador de Programas en Pascal, ya que podrán utilizar el lenguaje para realizar sus propios sistemas según lo requiera su carrera.

II.3.- SELECCIÓN DE SOFTWARE PARA EL DISEÑO DEL SISTEMA.

ANTECEDENTES DE LOS MANEJADORES DE BASES DE DATOS(DBASE-III, DBASE-IV, CLIPPER, VISUAL FOX PRO).

dBase

dBASE fue el primer sistema manejador de archivos usado ampliamente para microcomputadoras, publicado por Ashton-Tate para CP/M, y más tarde para Apple II, Apple Macintosh, UNIX [1], VMS [2], e IBM PC bajo DOS donde con su legendaria versión III Plus se convirtió en uno de los títulos de software más vendidos durante un buen número de años. dBASE nunca pudo superar exitosamente la transición a Microsoft Windows y terminó siendo desplazado por productos más nuevos como Paradox, Clipper, y FoxPro.

Incorporaba un lenguaje propio interpretado y requería un LAN PACK para funcionar sobre red local. En 1988 llegó finalmente la versión IV.

dBASE fue vendido a Borland en 1991. Al poco tiempo promovió una casi intrascendente versión 5, de la que llegó a haber versión para Windows. Luego vendió los derechos de la línea de productos en 1999 a dataBased Intelligence, Inc. (dBI) que sigue comercializando nuevas versiones, llamadas dBASE Plus, desde 1999

Durante la primera mitad de los '80s muchas otras compañías produjeron sus propios dialectos o variaciones del producto y lenguaje. Estos incluyeron FoxPro (ahora Visual FoxPro), Quick-Silver, Clipper, Xbase++, FlagShip, y Harbour. Todos ellos son llamados informalmente como **xBase** o **XBase**.

El formato subyacente de dBASE, el archivo **dbf**, es ampliamente utilizado en muchas otras aplicaciones que necesitan un formato simple para almacenar datos estructurados.

dBASE fue licenciado a los usuarios por un plazo de quince años basado en el inconcebible evento de que un usuario utilizara su copia de dBASE por tan largo período de tiempo.

Historia

La historia de dBASE empezó a mediados de 1960 como un sistema llamado RETRIEVE. Este sistema era usado, entre otros, por el Jet Propulsion Laboratory, que comisionó el desarrollo de su propia versión de RETRIEVE a uno de sus programadores, Jeb Long. El resultado fue un sistema llamado JPLDIS, que corría en la UNIVAC 1108 y estaba escrito en FORTRAN.

El creador original de dBase fue Wayne Ratliff [3]. En 1978, durante su estancia en el Jet Propulsion Laboratory (Laboratorio de Propulsión a Chorro) Ratliff escribió un programa de base de datos en assembler para ordenadores con sistema operativo CP/M para ayudar con las apuestas de fútbol en la oficina. Lo denominó **Vulcan** en referencia al personaje Mr. Spock de Star Trek. Se basó en el JPLDIS (Jet Propulsion Laboratory Display Information System)

desarrollado por Jeb Long. Más tarde lo usó para preparar sus impuestos y decidió que tenía un potencial comercial.

Las primeras copias tenían un precio de US$7000. La respuesta comercial fue baja y finalmente este hecho, sumado al estrés de las extensiones y las mejoras al sistema, hizo que su mercadeo se estancara.

[editar] Ashton-Tate

Un cliente de Vulcan se comunicó con George Tate y Hal Lashlee, a la sazón dueños de **Discount Software** y estos fueron a ver a Ratliff y su demostración de Vulcan. Impresionados, le hicieron un ofrecimiento de derechos exclusivos de mercadeo, que Ratliff aceptó. Finalmente la compañía creció lo suficiente como para contratar a Ratliff como vicepresidente de nuevas tecnologías, y luego como líder del proyecto dBASE.

Igualmente se vinculó a la compañía el programador original de RETRIEVE, Jeb Long, quien terminó por crear el lenguaje de programación interno de dBASE y quien fuera conocido como el gurú de los productos dBASE en Ashton-Tate.

dBase II

Vulcan es portado al IMSAI 8080, se le renombra a dBase II (Tate considera que una *versión 2'* dará una imagen de más seriedad y producto más elaborado) y de ahí a CP/M, donde se le añaden comandos de soporte de interfaz de video (en modo texto), y soporte de control de fujo (como DO WHILE/ENDDO) y lógica condicional (como IF/ENDIF). Para el manejo de datos, dBase proporciona detallados comandos procedurales y funciones para abrir y navegar por las tablas (como USE, SKIP, GO TOP, GO BOTTOM, y GO recno), manipula valores en los campos (REPLACE y STORE),y manipulación de Strings (como STR() and SUBSTR()), Fechas y Números. su habilidad para simultáneamente abrir y manipular múltiples ficheros conteniendo datos relacionados hará que Ashton-Tate lo califique de base de datos relacional aunque no cumpla con los criterios definidos por el Dr. Edgar F. Codd.

Alcanza un gran éxito, y se incluye en los paquetes de soft distribuidos con el Osborne I, la gama de ordenadores Kaypro y otros equipos. El nacimiento de los ordenadores domésticos hacen que se utilice para crear programas *profesionales* en equipos como las gamas Amstrad CPC y Amstrad PCW, el Commodore 128 y los equipos MSX con unidad de disco (el MSX-DOS soporta los ejecutables CP/M 8080 y Z80).

[editar] dBASE III

Las versiones originales fueron escritas en lenguaje ensamblador, pero a medida que el programa creció se tomó la decisión de re-escribir el código en lenguaje C. El resultado fue que las máquinas recientes corrían bien el código, pero no así las antiguas. En paralelo, se siguieron vendiendo versiones dBASE II hasta que los problemas de desempeño del III se corrigieron a mediados de 1985.

Obtenido de "http://es.wikipedia.org/wiki/DBase"

Clipper (lenguaje de programación)

Clipper es un lenguaje de programación imperativo creado en 1985 por Nantucket Corporation. En un principio se pensó en que Clipper sería un compilador para el sistema gestor de bases de datos dBase III (de hecho las *versiones estacionales* de Nantucket incluían una etiqueta que lo indicaba así), pero con el tiempo el producto maduró, convirtiéndose en un lenguaje más poderoso que el original, no sólo por sus propias implementaciones sino también por las ampliaciones al lenguaje desarrolladas por terceros en C y Pascal, de los que va heredando características. Esto lo convierte en la herramienta líder de desarrollo bajo MS-DOS de aplicaciones relacionadas con bases de datos, sobre todo programas de contabilidad y facturación (SAGE-SP, líder del mercado español, lo usa para Contaplus y FacturaPlus), y agendas comerciales y programas de tarificación (aproximadamente el 80% de las compañías de seguros de España lo utilizaron en los programas para sus agentes).

A diferencia de otros lenguajes xBase, Clipper nunca contó con un modo intérprete similar al de dBase, y sus utilidades para manejo de base de datos se entregaban con el código fuente. Como muchos otros lenguajes xBase, Clipper en realidad no es un compilador puro, pues traduce el código fuente a P-code (también llamado pseudocódigo) que es interpretado por una máquina virtual. Este sistema es rápido pero no tanto como lo sería un compilador puro que generara ensamblador. Por ello siempre necesita tener los ficheros de runtime.

Las primeras versiones se denominan *versiones estacionales* por hacer referencia a una estación del año en sus nombres oficiales. Todas ellas se nominaban como **compiladores dBase**. Estas fueron :

- Nantucket Clipper Winter'84 - lanzada el 25 de mayo de 1985
- Nantucket Clipper Summer'85 - lanzada en 1985
- Nantucket Clipper Winter'85 - lanzada el 29 de enero de 1986
- Nantucket Clipper Autumn'86 - lanzada el 31 de octubre de 1986
- Nantucket Clipper Summer'87 - lanzada el 21 de diciembre de 1987

Clipper 5.0 supone un salto cualitativo del lenguaje, aunque comienza con mal pié. Dada la popularidad cada vez mayor (Summer 87 ha sido utilizada hasta el año 2000 como herramienta de desarrollo), se decide centrarse más en ampliar el lenguaje que en ser un *compilador mejorado* de dBase. Se implementan los pseudoobjetos y otras mejoras... pero el producto se lanza con numerosos bugs que hacen que el público objetivo se retraiga y siga usando Summer87, mucho más estable. La 5.01 corrige muchos de estos problemas, pero no será hasta la 5.2 que se produzca el vuelque masivo. Las versiones 5 de Nantucket son :

- Nantucket Clipper 5.00 - lanzada en 1990
- Nantucket Clipper 5.01 - lanzada el 15 de abril de 1991
- Nantucket Clipper 5.01 Rev.129 - lanzada el 31 de marzo de 1992

El gigante del soft Computer Associates adquiere Nantucket y se lanza a mejorar el producto afianzado las características heredadas de C, en particular el tipo de datos **code-block** (literalmente *bloque de código*, un híbrido entre las macros de dBase, la evaluación de cadenas de caracteres y los punteros de funciones). Otra de las mejoras procedentes de la 5.0 es el sistema de Replaceable Database Drivers (RDD o drivers reemplazables de base de datos), que permite con una sola sentencia cambiar entre diferentes normas de base de datos . La aparición de la versión 5.2, con una carrera frenética de subversiones (con mejoras y corrección de errores) hasta la 5.2c, marca el comienzo de la migración masiva de quienes todavía permanecían en Summer'87. Devendrá en la versión de Clipper más usada de la historia. Por el contrario su sucesora, 5.3, pese a implementar mejoras, cae en un error de bulto, al no tener en cuenta la compatibilidad con al menos las más populares librerías de Clipper (tanto comerciales como freeware), y al consumir más recursos de DOS.

- CA Clipper 5.01a -
- CA Clipper 5.20 - lanzada el 15 de febrero de 1993
- CA-Clipper 5.2a - lanzada el 15 de marzo de 1993
- CA Clipper 5.2b - lanzada el 25 de junio de 1993
- CA-Clipper 5.2c - lanzada el 6 de agosto de 1993
- CA Clipper 5.2d - lanzada el 25 de marzo de 1994
- CA-Clipper 5.2e - lanzada el 7 de febrero de 1995
- CA Clipper 5.30 - lanzada el 26 de junio de 1995
- CA Clipper 5.3a - lanzada el 20 de mayo de 1996
- CA Clipper 5.3b - lanzada el 20de mayo de 1997

Computer Associates decide abandonar Clipper ante la pujanza de Microsoft Windows, y vuelca parte del desarrollo de Clipper (el *proyecto Aspen* de Nantucket) en su nueva herramienta CA-Visual Objects, que se presenta casi a la vez que Clipper 5.3 Pero el abandono de la síntaxis xBase y el no proveer una herramienta de migración adecuada, unido al alto precio del producto (que además debía competir con otros productos de la propia casa, uno de ellos basado en BASIC), hace que el grueso de programadores Clipper opten por permanecer en 5.2/5.3 con librerías de terceros como FiveWin o migren a herramientas xBase como Visual FoxPro a medida que el mercado DOS va reduciéndose.

El 22 de abril de 2002 Computer Associates y GrafX Software anuncian que han alcanzado un acuerdo de licenciamiento, marketing y desarrollo de dos de sus lenguajes de desarrollo : CA-Clipper y CA-Visual Objects.

Una de las principales características que ayudó al éxito de Clipper fue la posibilidad de expandir el lenguaje con rutinas en C y ensablador. Varias de ellas, como CodeBase o Apollo son RDDs. Con la aparición de Windows se desarrollaron varias para portar las aplicaciones Clipper a Windows. De ellas la más popular es la española FiveWin (http://www.fivetechsoft.com), empleada en los productos líderes de contabilidad en España.

Además, el uso de linkadores alternativos permitieron mejorar el rendimiento del EXE generado. El más aclamado es Blinker, que añade un extensor de

DOS con modo protegido (es utilizado con numerosos lenguajes y compiladores). Añadió soporte para compilar programas y librerías para Windows.

En la actualidad el lenguaje Clipper está siendo activamente implementado y extendido por varios proyectos y vendedores. Entre los proyectos de software libre podemos destacar Clip, Harbour y xHarbour. Entre los compiladores comerciales *Xbase++* y *FlagShip*.

Varias de esas implementaciones son portables gracias a su desarrollo en C (DOS, Windows, Linux (32 y 64 bits), Unix (32 y 64 bits), y Mac OS X), soportando varias extensiones del lenguaje [1], cuentan con varias extensiones del lenguaje, y varios Replaceable Database Drivers (RDD) que soportan los formatos más populares, como DBF, DBTNTX, DBFCDX (FoxPro y Comix), MachSix (Apollo), SQL, y más. Todas estas nuevas implementaciones mantienen la completa compatibilidad con la síntasix estándar xBase, a la vez que ofrecen programación orientada a objetos y síntaxis orientada al destino como SQLExecute().

En 2005, los newsgroups de Usenet relativos a Clipper comp.lang.clipper y comp.lang.clipper.visual-objects siguen activos.

Enlaces externos

- FiveTech] desarrolladores de FiveWin y uno de los motores del Proyecto Harbour
- Proyecto Harbour
- ViaOpen desarrolladores de controles para FiveWin y FiveWin Pocket PC

Obtenido de "http://es.wikipedia.org/wiki/Clipper_%28lenguaje_de_programaci%C3%B3n%29"

Visual FoxPro

Visual FoxPro es un lenguaje de programación orientado a objetos y procedural, un Sistema Gestor de Bases de datos o Database Management System (DBMS), y desde la versión 7.0, un Sistema administrador de bases de datos relacionales, producido por Microsoft.

Características

Visual FoxPro ofrece a los desarrolladores un conjunto de herramientas para crear aplicaciones de bases de datos para el escritorio, entornos cliente/servidor, tablet PC o para la Web.

Entre sus características se pueden enumerar:

- Capacidades poderosas y muy veloces para el manejo de datos nativos y remotos.
- Flexibilidad para crear todo tipo de soluciones de bases de datos.
- Lenguaje de programación Orientado a objetos.
- Utilización de sentencias SQL en forma nativa.
- Poderoso manejo de vistas y cursores y control completo de estructuras relacionales.
- Su propio gestor de base de datos incorporado. Sin embargo, también puede conectarse con servidores de base de datos, tales como Oracle, Microsoft SQL Server o MySQL.
- Cuenta con un motor de generación de informes renovado y muy flexible para soluciones más robustas.
- Desde la versión 9.0, amplio soporte de XML, tanto como fuente de datos (por ej., servicios Web basados en XML) como por generar reports en formato XLM.
- Desde la versión 7.0, soporte de la tecnología IntelliSense de Microsoft

La última versión liberada es la 9.0. La próxima versión, 'Sedna', será un poderoso y completo lenguaje que permitirá al producto interactuar aun más con VisualStudio.net, SQLServer2005, SQLExpress2005 y Office12, Windows Vista.

La versión 9.0 de Visual FoxPro cuenta con el SP1 en la que hay algunas nuevas características y especialmente brinda estabilidad al producto.

[1]

Historia

Visual FoxPro proviene de FoxPro, que a su vez deriva de FoxBASE, creado por Fox Technologies en 1984; inicialmente un compilador de dBase, acabó superándolo y con Clipper, convirtiéndose en una de las estrellas de los lenguajes xBase. Fox Technologies fue adquirido por Microsoft en 1992.

Visual FoxPro 3.0, fue la primera versión “Visual”, redujo su compatibilidad a solo Mac y Windows (La última versión de FoxPro (2.6) corría en MS-DOS, MS Windows, Mac OS y UNIX), versiones posteriores fueron solo para Windows. La versión actual se basa en archivos COM y Microsoft ha declarado que no piensan crear una versión .NET.

En la versión 5.0 se integra en Microsoft Visual Studio añadiéndosele el soporte de Microsoft Source Safe. Hasta entoces es visto típicamente por el publico como meramente un Sistema de gestión de base de datos (SGBD), ignorando

[1] *José A. Ármalo.* Clipper Avanzado. Editorial McGraw Hill.

el hecho de que no solo incluye el entorno SGBD, sino un completo lenguaje de programación.

Visual FoxPro 6.0, publicado en 1999, no supone un cambio radical respecto de la anterior versión sino únicamente una mejora en sus diversas funcionalidades y una adaptación al mundo internet y al mundo de los objetos. Esta versión hace más atractivo a los desarrolladores el tratamiento de los datos en los entornos COM. Es un paso más en la evolución de este producto desde un entorno de aplicaciones monousuario o de redes pequeñas centradas en los datos hacia una herramienta orientada a objeto diseñada para la construcción de la lógica del negocio en los entornos multi-tier con una fuerte orientación hacia los tratamientos intensivos de datos en Internet. Pese a su relativa antigüedad, es hoy todavía ampliamente utilizado en grandes empresas (por ej., la compañía de seguros Mapfre) por su estabilidad.

Visual FoxPro 7.0, publicado en 2001, supuso su salida de Visual Studio, pues aunque en un principio se pensaba incluir a Fox en .NET, no era posible sin romper con la herencia de anteriores versiones. Esta versión incorporó por primera vez el IntelliSense, y se mejoró el manejo de arrays, acercándolo al de cursores.

A finales del 2002, algunos miembros de comunidades demostraron que Visual FoxPro puede correr en Linux usando el emulador de Windows Wine. En el 2003, esto llevo a quejas de Microsoft: se dijo que el desarrollo de código de FoxPro para rutinas en máquinas no-Windows viola el Acuerdo de Licencia de Usuario Final.

Los rumores de que Microsoft planea terminar el soporte para FoxPro han sido comunes desde su adquisición del producto, a pesar del hecho de que éste ha tenido el tiempo de vida de soporte más largo para un producto de Microsoft (hasta el 2014). VFP 9 fue lanzado el 17 de diciembre del 2004 y el equipo de Fox está trabajando actualmente en un proyecto cuyo nombre clave es Sedna que será construido sobre el código base de VFP 9 y consistirá principalmente en componentes Xbase que soportarán un número de escenarios interoperables con varias tecnologías de Microsoft incluyendo SQL Server 2005, .NET, WinFX, Windows Vista y Office 12.

No obstate, siempre parece el *patito feo* de los productos Microsoft. Solicitar información sobre él en cualquier stand oficial de una feria informática como el español SIMO supone que deban preguntar a al menos 3 personas, y muchas veces el usuario de Fox disponga de mayor información que los empleados (que no azafatas) presentes en él. Son varios los testimonios de visitas a empresas por parte de delegaciones de la central de Microsoft que no han sabido reconocer el producto como propio o lo han confundido con Visual Basic.

C a p í t u l o iii

Diseño del sistema

CAPÍTULO III. DISEÑO

GENERADOR DE PROGRAMAS EN PASCAL

III.1.- ANTECEDENTES

El Generador de Programas en Pascal, es un software desarrollado en Visual Fox Pro, cuyo objetivo primordial es construir programas de computadora en lenguaje de programación PASCAL, a partir de un pseudocódigo, esta herramienta fue desarrollada con el propósito de ayudar a los estudiantes de quinto y sexto semestre que cursan la materia de Cibernética y Computación I y II en la Escuela Nacional de Ciencias y Humanidades

La idea de desarrollar el Generador de Programas en Pascal nace como una inquietud de proporcionar una herramienta de programación fácil de usar que permita ser utilizada por los estudiantes del Colegio como una interfase entre las unidades 3 y 4 de la materias de Cibernética y Computación I y como un soporte de programación para el curso de Cibernética y Computación II, la inquietud anterior surge al poder constatar en repetidas ocasiones a lo largo de mi experiencia docente, que el alumno del Colegio cuando estudia la unidad 4 (Introducción a la programación), al momento de solicitarle que elabore el programa de computadora en lenguaje Pascal de un problema ya resuelto en la Unidad 3 (metodología de solución de problemas) y donde se pide que tome como referencia la solución lógica desarrollada anteriormente y donde hizo uso de un pseudocódigo o diagrama de flujo el alumno no recuerda lo estudiado.

La situación anterior provoca a mi manera de ver las situaciones que a continuación expongo:

- Que si el profesor desea que el alumno recuerda la solución lógica del problema planteado, de nueva cuenta retome los contenidos temáticos de la unidad 3 y vuelva a explicar la Metodología en la solución de problemas.

- O bien que el profesor explique los conceptos de programación del lenguaje Pascal sin tomar en cuenta la solución lógica del problema, situación que crea en el alumno mal hábito, en virtud de que provoca que el estudiante del Colegio se acostumbre a construir programas de computadora en Lenguaje Pascal sin tener como soporte la solución lógica.

Con base a lo anteriormente expuesto me dedique a la tarea de trabajar en un software de computadora que permitiera al estudiante del Colegio poder relacionar los contenido temáticos de las unidades 3 y 4 del curso de Cibernética y Computación I, además de proporcionar herramientas sólidas para el curso de Cibernética y Computación II, en virtud de que estoy plenamente convencida de que para poder tener éxito en la construcción de

programas de computadora en cualquier lenguaje de programación forzosamente se debe de partir de la solución lógica elaborada a través de un pseudocódigo y/o diagrama de flujo.

Por otra parte quisiera mencionar que el mal hábito de construir programas de computadora sin tomar en cuenta la solución lógica se ha difundido en virtud de que la inmensa mayoría de los profesores que imparten la materia no retoman por falta de tiempo los contenidos temáticos de la unidad 3 del curso de Cibernética y Computación I. La situación anterior se agrava aún más en los estudiantes que egresan de la Escuela Nacional de Ciencias y Humanidades y que optan por cursar las carreras de Ciencias e Ingeniería en virtud de que existen estudiantes de los últimos semestres de la carreras antes mencionadas que tienen serias deficiencias en la elaboración de programas de computadora ya que nunca se les inculco que para tener éxito en su construcción primeramente se debería desarrollar su solución lógica.

Asimismo deseo manifestar que con el propósito de optimizar la herramienta desarrollada, esta a sido probada con estudiantes del quinto semestre de la materia de Cibernética y Computación I de los grupos 556 y 557, los alumnos en un principio se preguntaban si la presente herramienta les brindaría un beneficio, más sin embargo han observado que el uso del Software desarrollado les a permitido verificar la similitud que existe en la elaboración del pseudocódigo y el lenguaje de programación, asimismo por la ayuda en línea que posee el software le ha permitido al estudiante adentrarse en el estudio de las herramientas computacionales del lenguaje Pascal, **a continuación me permito presentar una encuesta levantada con los estudiantes de los citados grupos donde se muestran sus comentarios**.

III.2.- CARÁCTERÍSTICAS TÉCNICAS

El Generador de Programas en Pascal un software desarrollado en el paquete: Visual Fox Pro Versión 6.0 , corre en máquinas con Windows desde la versión 97, requiere un mínimo de espacio en disco menos de 100 Mb y una resolución de monitor VGA, es necesario el software Microsoft Word y el paquete de programación Turbo Pascal, que tampoco implica mayor problema, ya que todas estas herramientas se encuentran instaladas en los equipos de cómputo del Colegio de Ciencias y Humanidades Plantel Azcapotzalco.

III.3.- FASE DE ANÁLISIS

El Generador de Programas en Pascal es una herramienta que funciona teniendo como base un lenguaje Pseudocódigo, en infinidad de ocasiones he observado la manera en que se obtiene la solución lógica de un problema a través de este, en múltiples ocasiones me ha surgido la inquietud de cómo construir un programa de computadora teniendo como referencia el Pseudocódigo, a continuación me permito exponer la fase de análisis:

- En un lenguaje Pseudocódigo se utilizan verbos tales como “ACEPTO”, “IMPRIME”, “REPITE”, o bien acciones como “ASIGNACIÓN”, “PREGUNTA”, “FIN DE REPITE” ó “FIN DE SÍ”, con la ayuda de estos se resuelven una inmensa mayoría de problemas, junto a estas acciones intervienen otros elementos como son las “VARIABLES”, “CONSTANTES”, y operaciones aritméticas tales como “SUMA, RESTA, MULTIPLICACIÓN” ó DIVISIÓN”, PORCENTAJE, RAIZ CUADRADA”.

- Cuando utilizas un lenguaje Pseudocódigo para encontrar la solución lógica de un problema, interrelacionas los elementos anteriores para construir las diferentes acciones que permiten construir la solución del problema, uno de las dificultades más importantes que enfrente es como diferenciar un verbo de una variable o constante, para solucionar este situación emplee un analizador de instrucciones que tuvo la capacidad de revisar carácter a carácter una línea, el analizador debería identificar en que momento construyo uno palabra y tener la capacidad de diferenciar un verbo de una variable o constante.

- El analizador debería revisar líneas de hasta 80 caracteres, los caracteres permitidos podrían ser cualquiera (letras, números o símbolos especiales), en el momento de armar una palabra debería de reconocer si se trataba de un verbo o de una variable. En el caso de que se tratara de un verbo debería dejarlo pasar sin efectuar acción alguna, en el caso de ser una variable debería de asignarla a un banco de variables, en el caso de detectar un signo especial como una operación aritmética, asignación o igual lo debería dejar pasar.

- En resumen al terminar de actuar el analizador de caracteres tenía la función de dejar un banco de variables y un banco de instrucciones con el lenguaje pseudocódigo.

- Cuando el banco de variables se había generado debía de asignarle su tipo de dato, el cual podría ser string, integer o real, así como identificarle un tipo de acción a desarrollar, la cual podría ser de entrada, proceso o salida.

- El siguiente paso consistía en construir las primeras instrucciones de un programa en Pascal, definiendo el Nombre del programa, dispositivos a utilizar y definir el área de variables, en donde se les

asignaría el tipo de dato determinada de cada una de las variables que se encontraban en el banco respectivo.

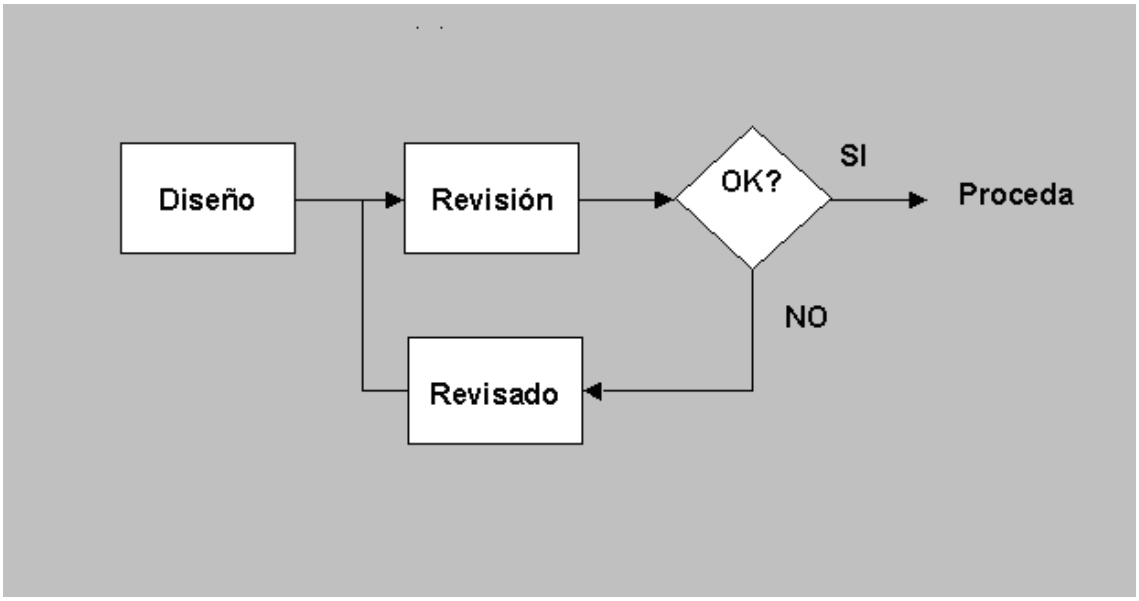
- A continuación había que construir el cuerpo del programa, para ello debería de analizar si existía alguna indicación de definir algún o algunos acumuladores o existía alguna indicación de generar algún ciclo.

- Posteriormente debía de analizar el banco de variables, la razón de ello es que debía de verificar si era necesario realizar alguna acción como alguna entrada de datos, situación que verificaba al detectar que la variable era de entrada.

- La siguiente acción a realizar era revisar el banco de instrucciones que era la parte donde se habían definido las indicaciones de Pseudocódigo, esta parte era transportada al cuerpo del programa, de manera similar como el usuario del programa la había escrito.

- La última parte del programa consistía en analizar de nueva cuenta el banco de variables, la razón de ello se debía a que tenía la necesidad de verificar si alguna de las variables seleccionadas eran de salida, en el caso de que esto sucediera se debería de generar una instrucción de salida.

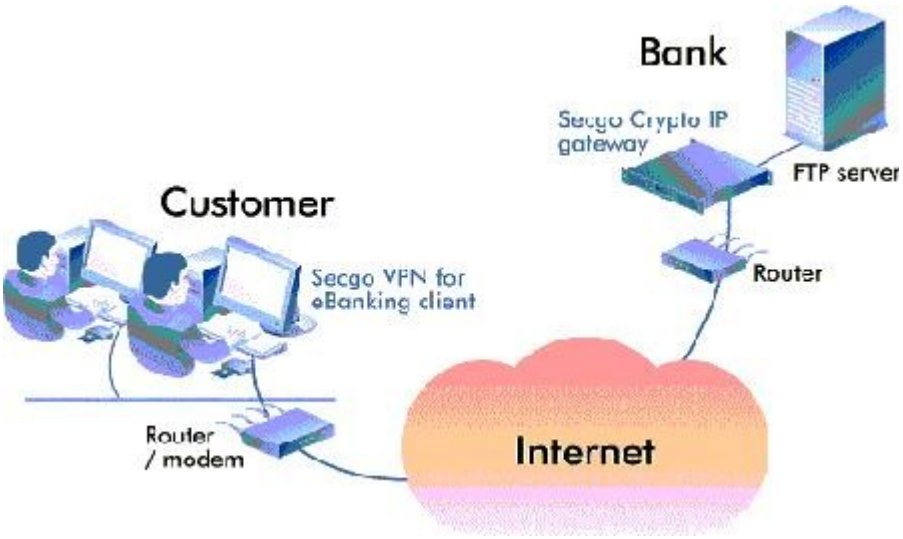
- Por último se debería de generar una indicación de que el programa había terminado, para ello debería de construir una instrucción de Fin.



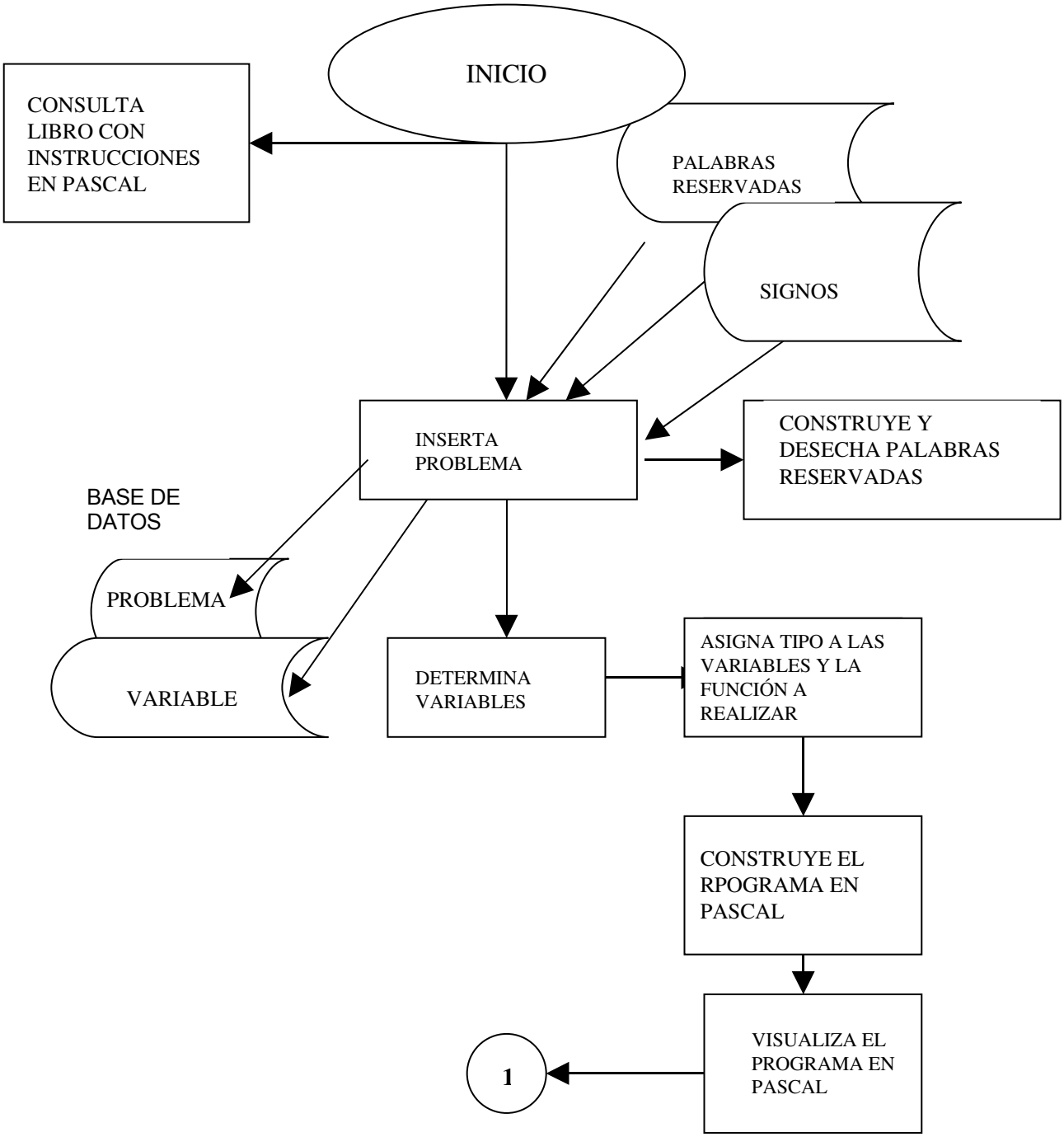
III.4.- FASE DE DISEÑO

La fase de diseño consistió en realizar en Visual Fox Pro las indicaciones anteriormente definidas.

Para el logro de las acciones anteriormente comentadas, fue necesario definir la organización y las estructuras de la Base de Datos a emplear.



III.5 DIAGRAMA DE BLOQUES
"GENERADOR DE PROGRAMAS EN PASCAL"



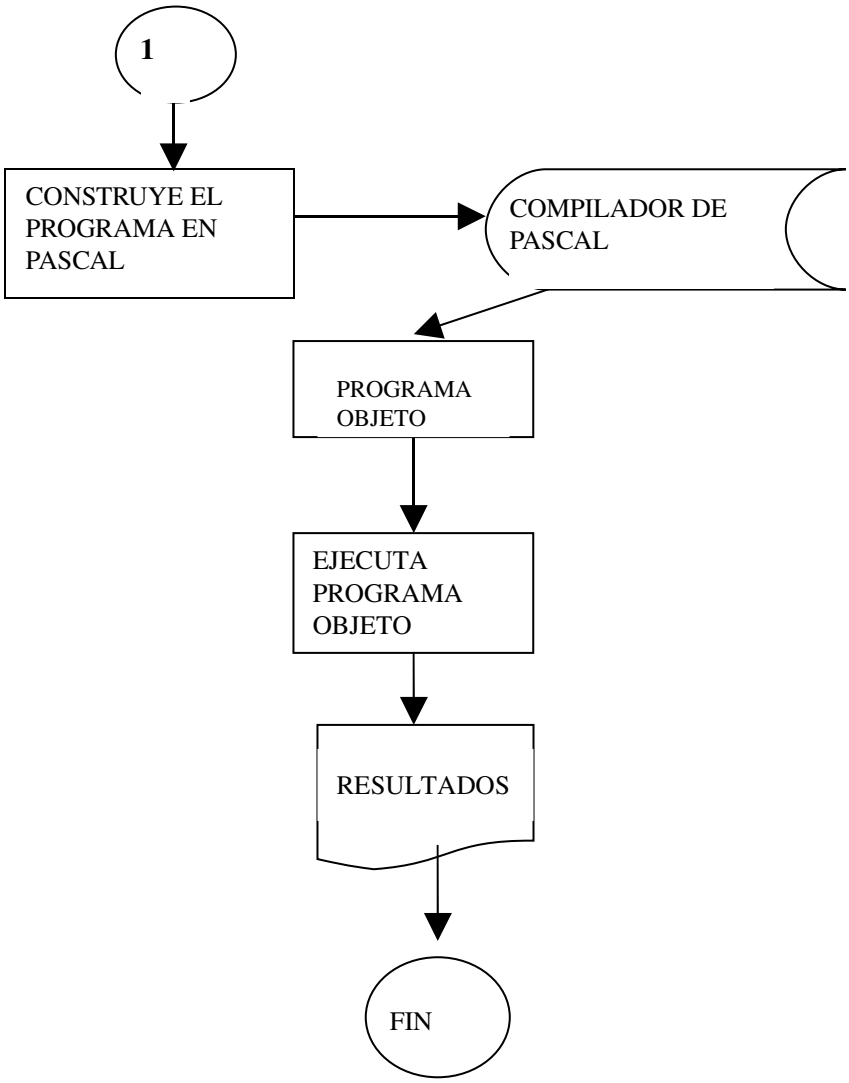


Fig. 3.1 Representación del diagrama de bloques del "Generador de Programas en Pascal"

MÉTODO DE RUTA CRÍTICA
Número de Actividades
6

ID	ACTIVIDADES	DURACIÓN	PROCEDENCIA
		semanas	
1	Análisis del sistema	3	
2	Diseño del sistema	4	1
3	Programación del sistema	4	2
4	Puesta a punto del sistema	3	3
5	Pruebas del sistema	4	4
6	Liberación del sistema	3	5

Fig. 3.2 Representación de la Ruta Crítica del Sistema "Generador de Programas en Pascal"

Diagrama Entidad-Relación (E-R)

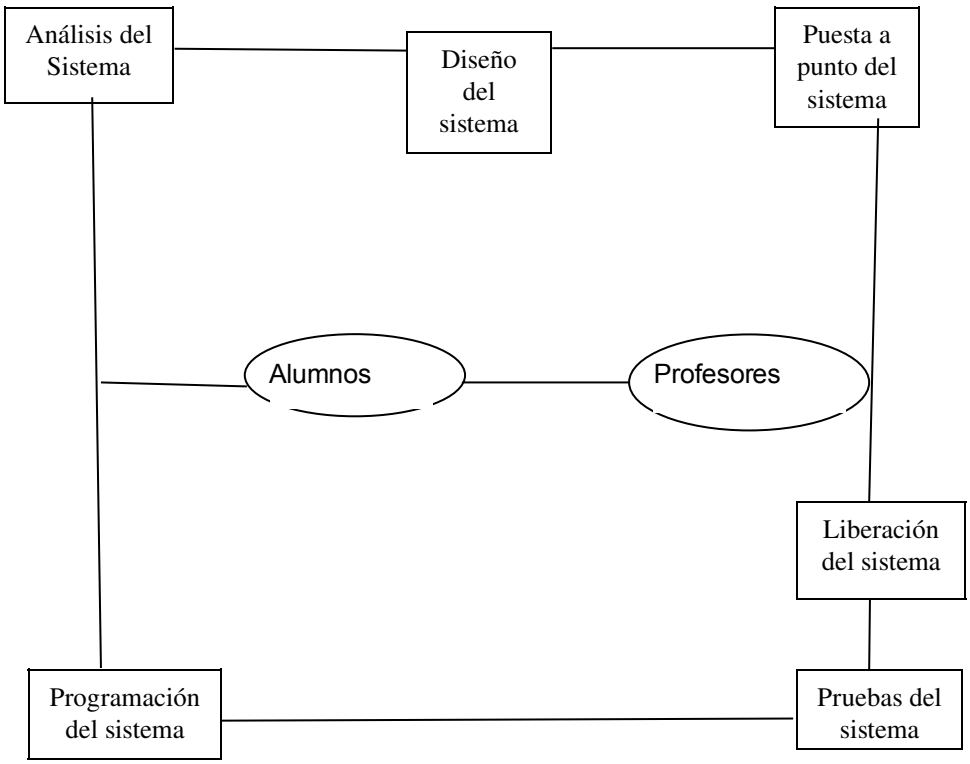


Fig. 3.3 Diagrama Entidad-Relación del Sistema "Generador de Programas en Pascal"

Ruta Crítica

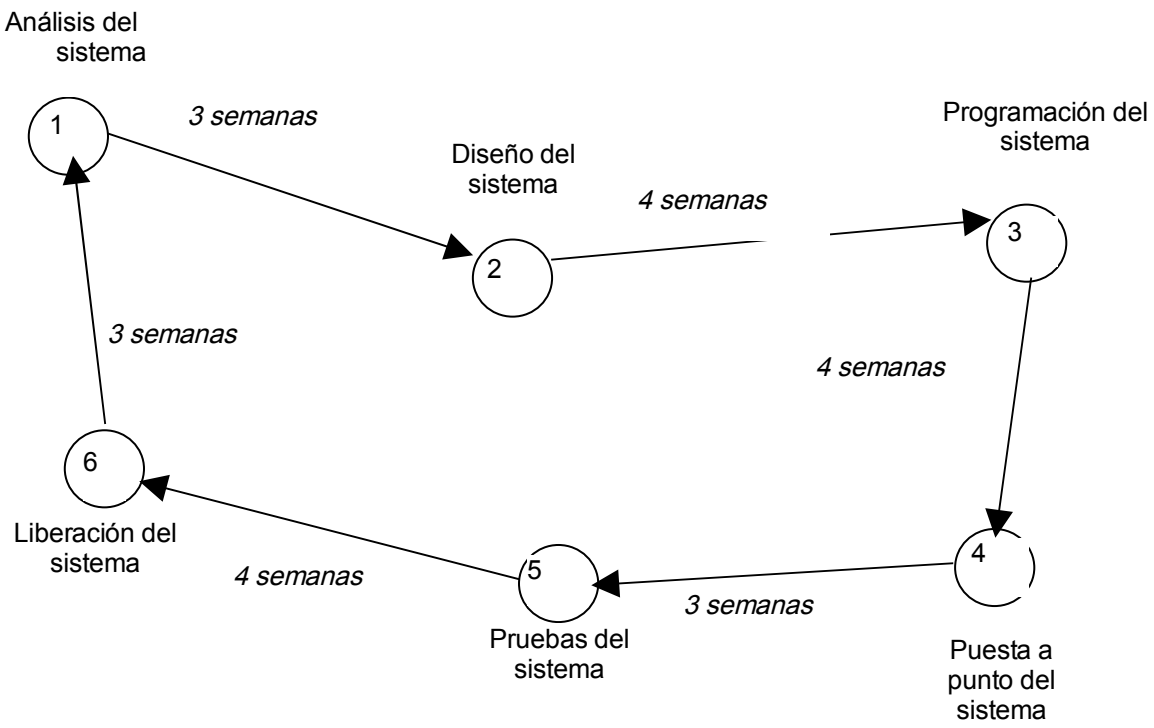


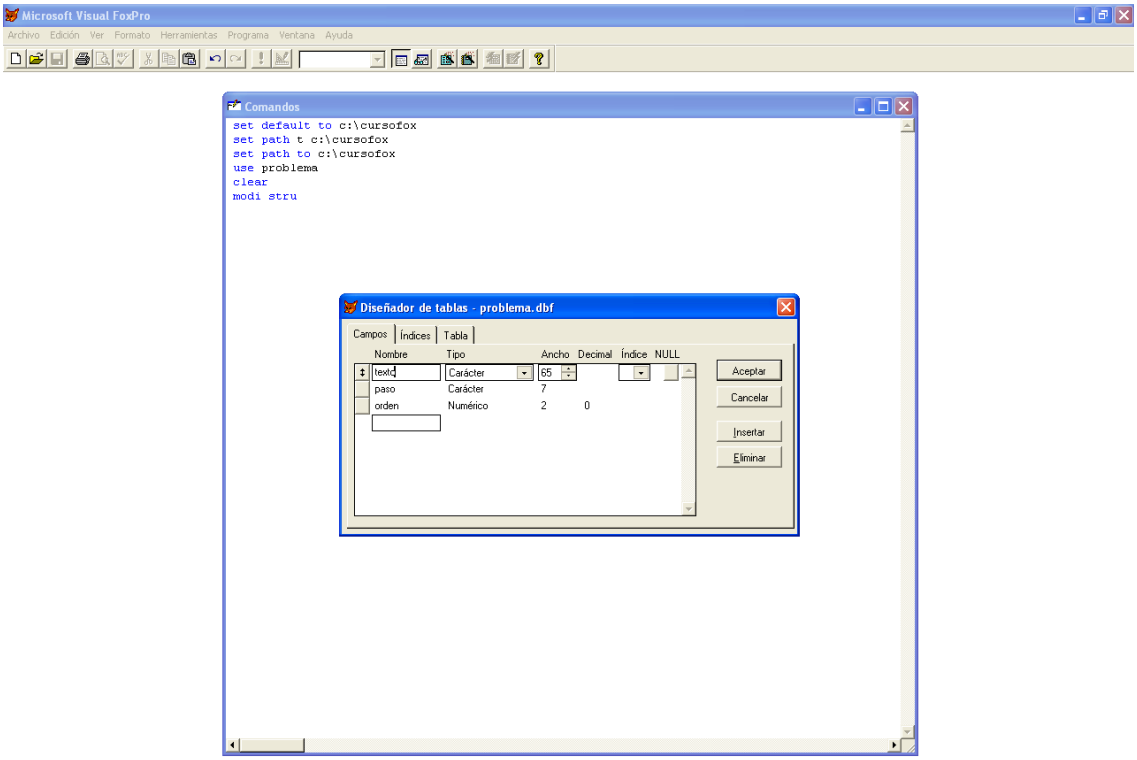
Fig. 3.4 Representación de la Ruta Crítica en diagrama Global con tiempos definidos, del "generador de Programas en Pascal"

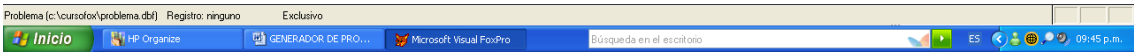
III.6 DEFINICIÓN DE TABLAS

- TABLA PROBLEMA

Objetivo.- Contener la definición original del Pseudocódigo, tal como el usuario del software lo ha digitado.

Campos





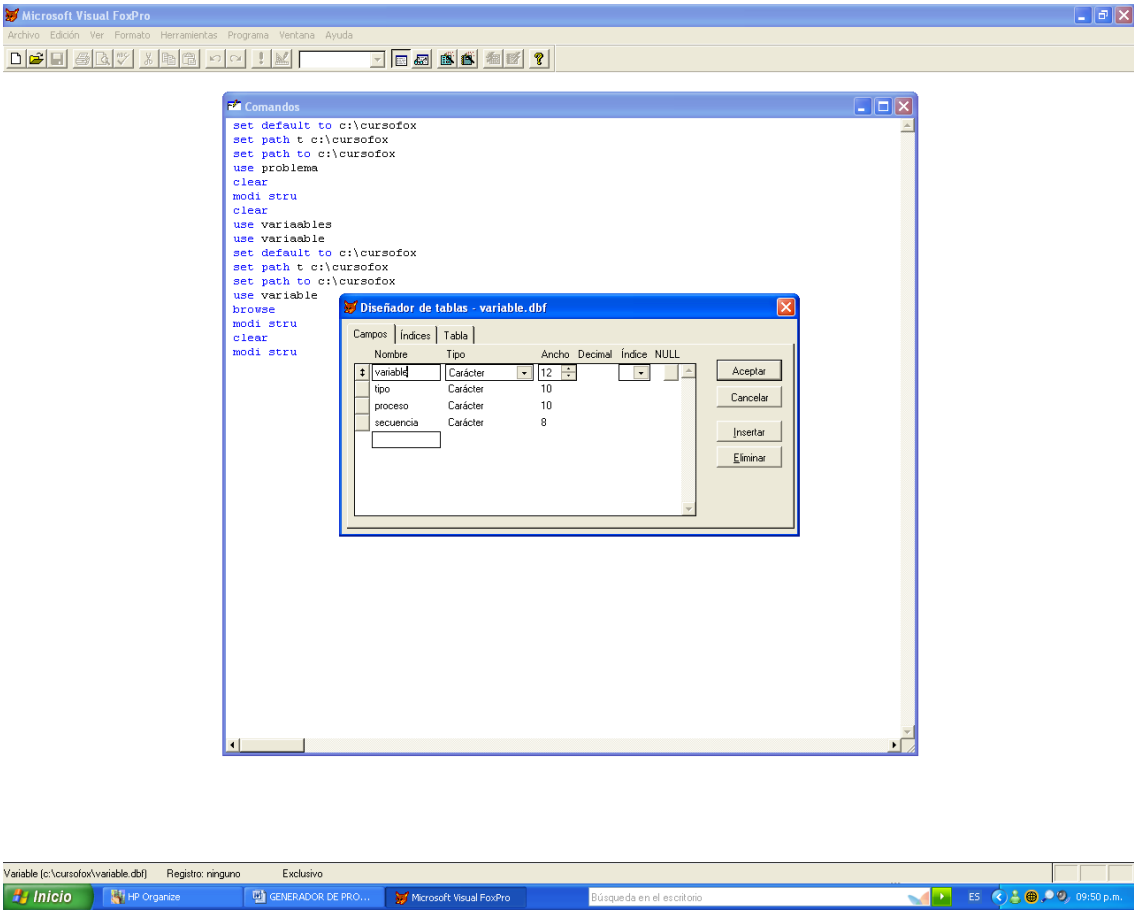
Definición de Campos

Nombre del Campo	Función
TEXTO	
PASO	
ORDEN	

Organización : Secuencial

- **TABLA VARIABLES**

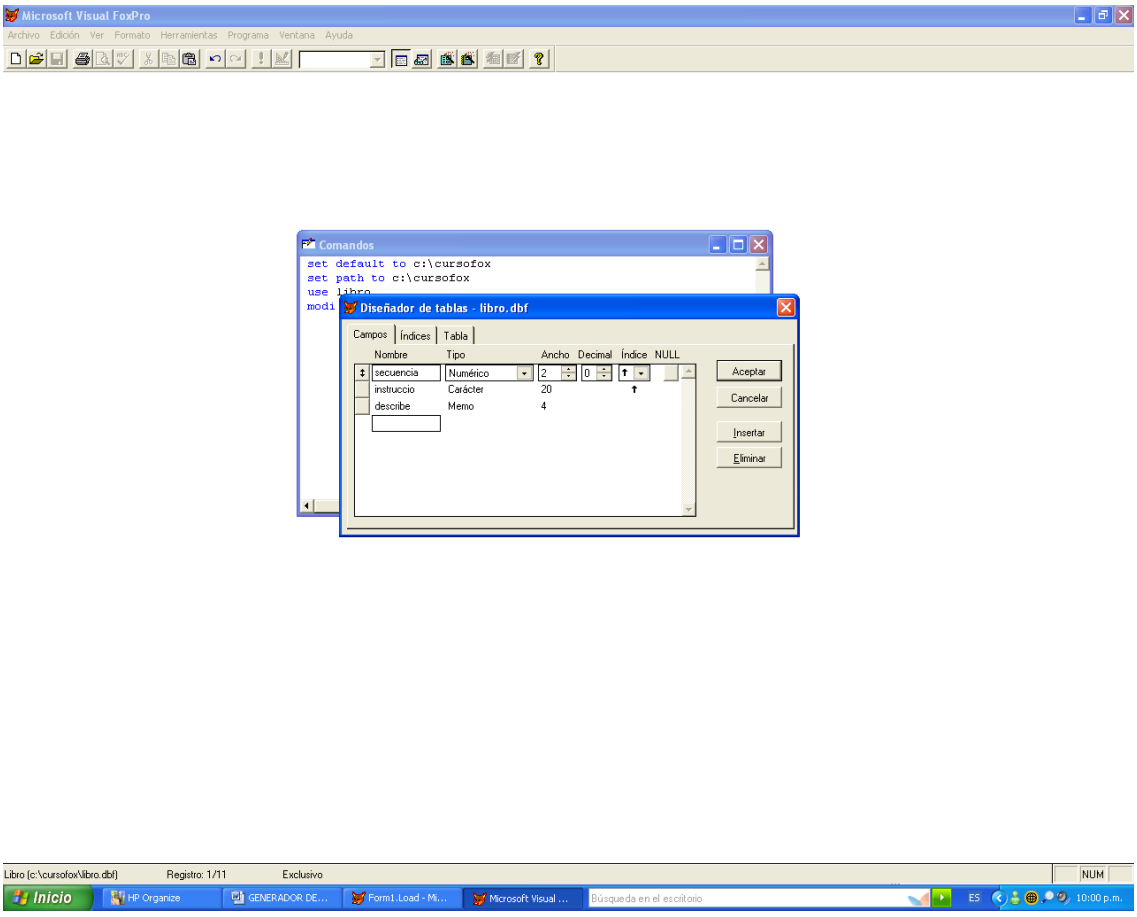
Objetivo.- Contener el total de variables que emplea el Pseudocódigo.



Nombre del Campo	Función
VARIABLE	
TIPO	
PROCESO	
SECUENCIA	

Organización : Indexada
Campo Índice : Variable

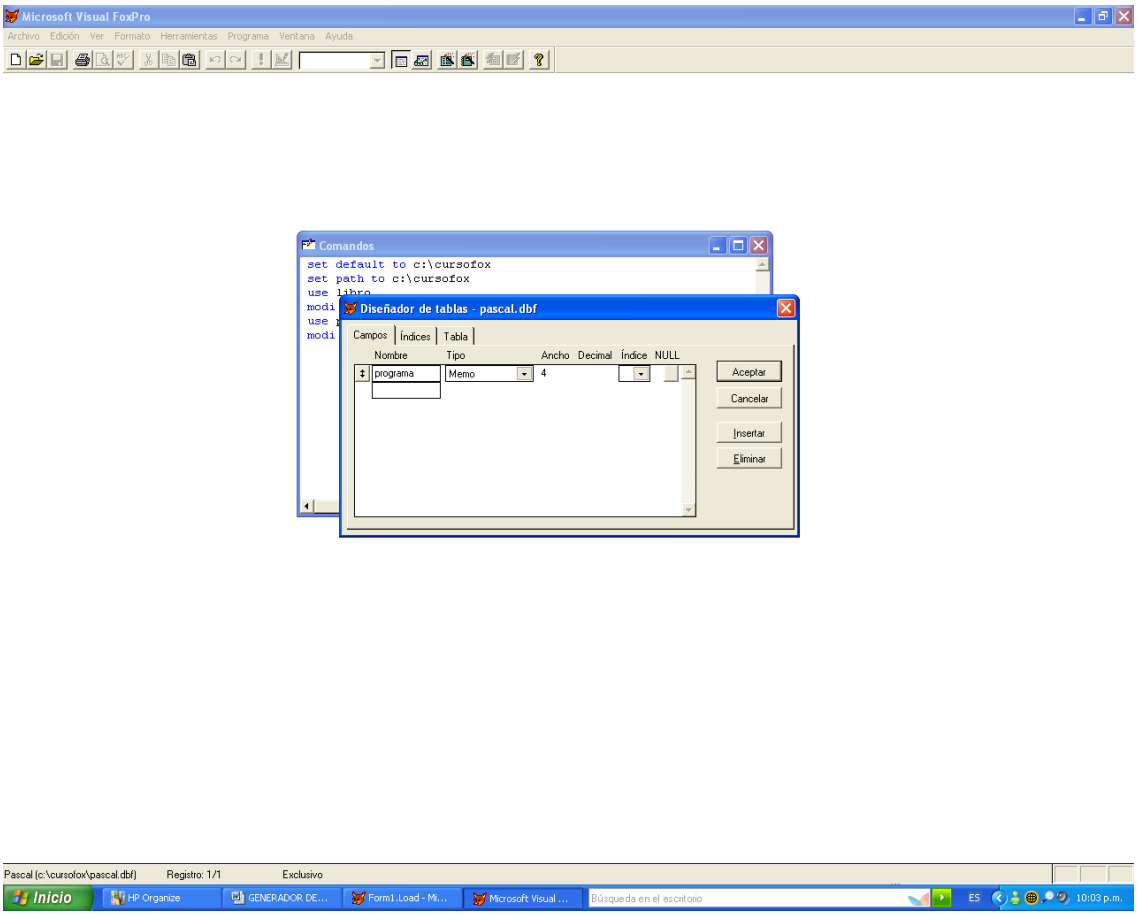
- **TABLA LIBRO**



Nombre del Campo	Función
SECUENCIA	
INSTRUCCIÓN	
DESCRIBE	

Organización : Indexada
Campo Índice : Secuencia

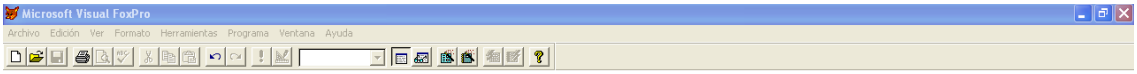
- **TABLA PASCAL**

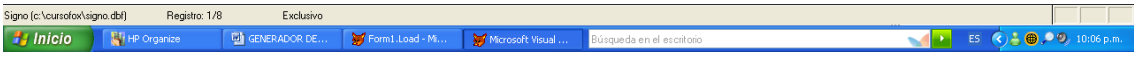


Nombre del Campo	Función
PROGRAMA	

Organización : Secuencial

- **TABLA SIGNO**

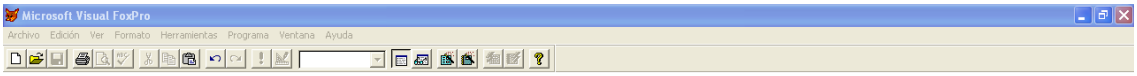


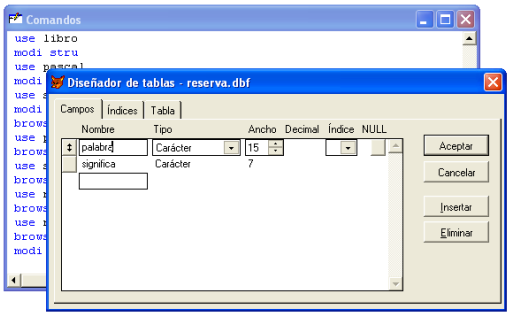


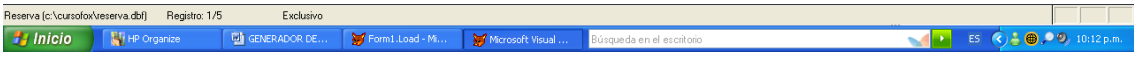
Nombre del Campo	Función
SIGNO	
DESCRIBE	

Organización : Indexada
Campo Índice : Signo

- **TABLA RESERVA**





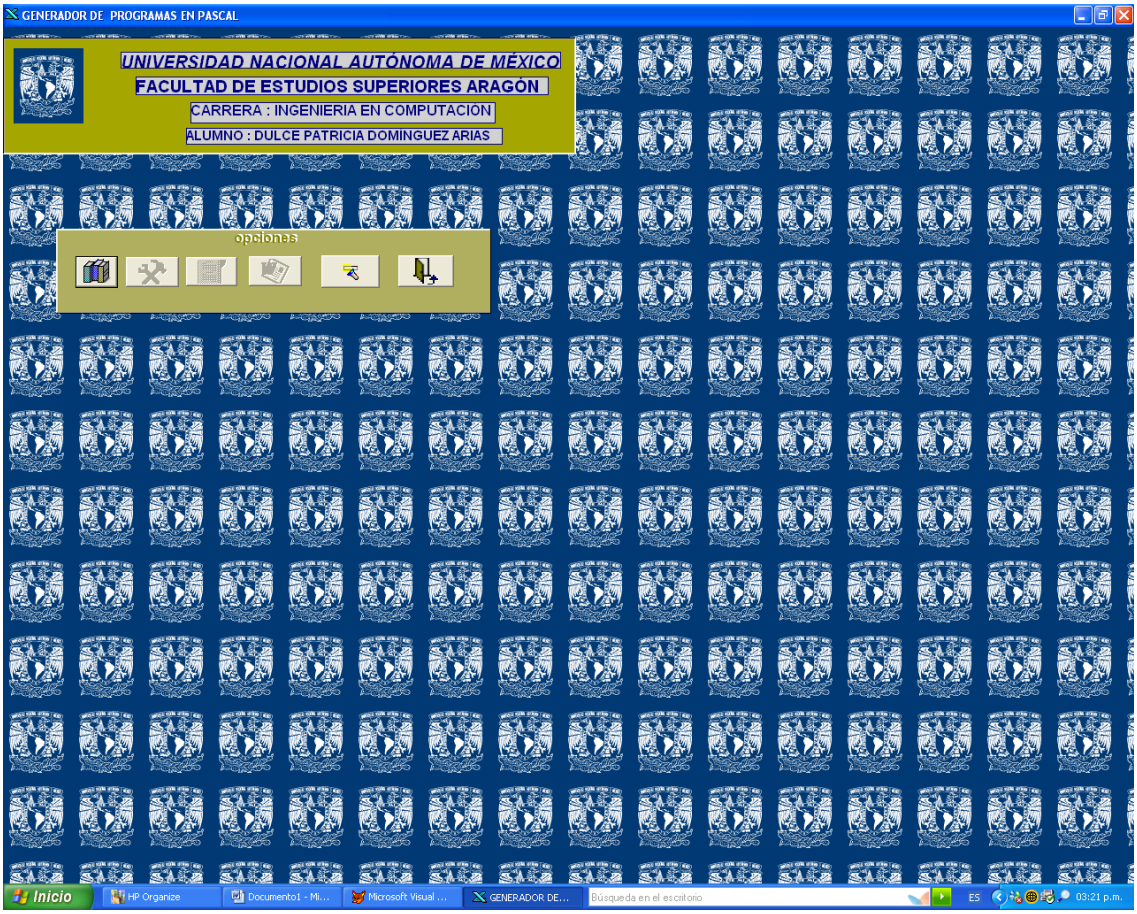


Nombre del Campo	Función
PALABRA	
SIGNIFICA	

Organización : Indexada
Campo Índice : Palabra

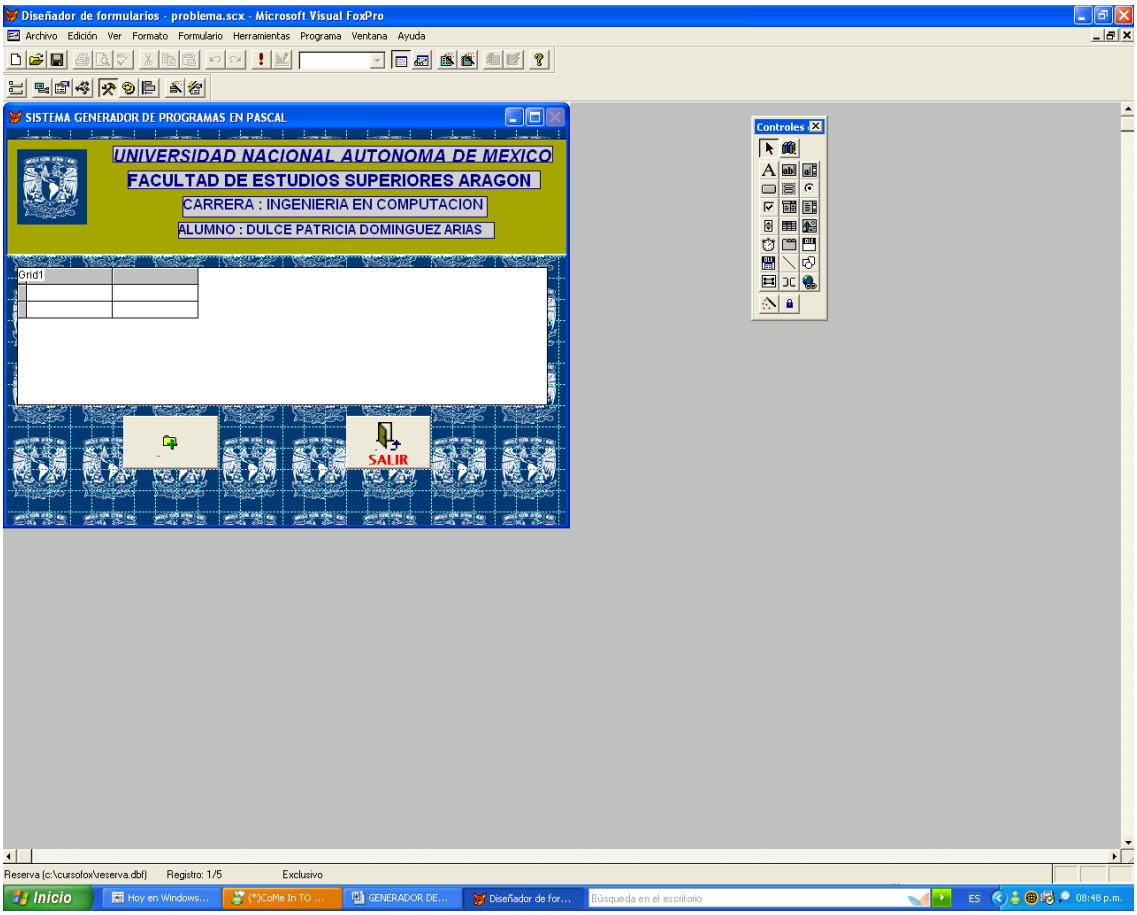
III.7 DEFINICIÓN DE LA OPERACIÓN DEL GENERADOR MÓDULOS DEL PROGRAMA

La operación del Generador de programas en Pascal, se divide en módulos, al ejecutarse el programa nos presenta la siguiente pantalla.



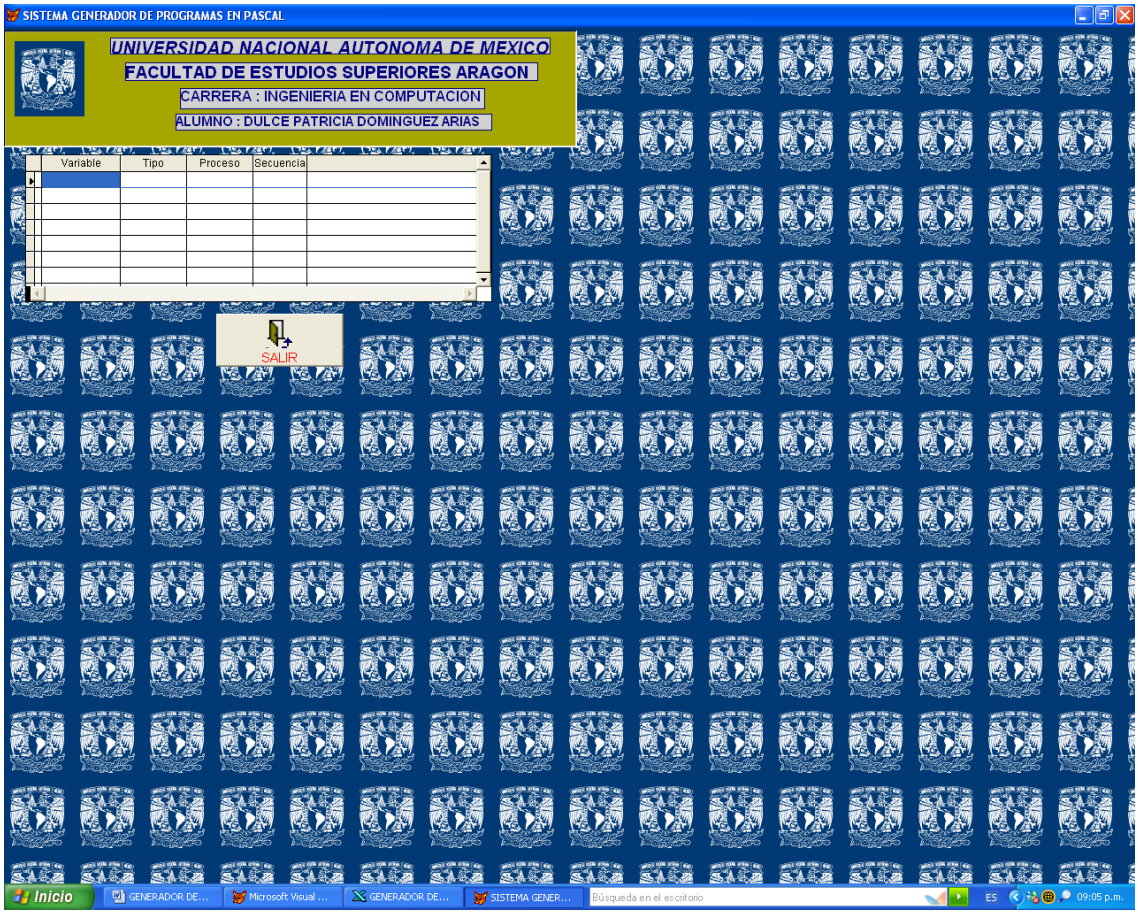
- **Módulo Problema**

Permite interactuar al usuario con el Generador de Programas, al invocar este icono, se presenta la siguiente pantalla.



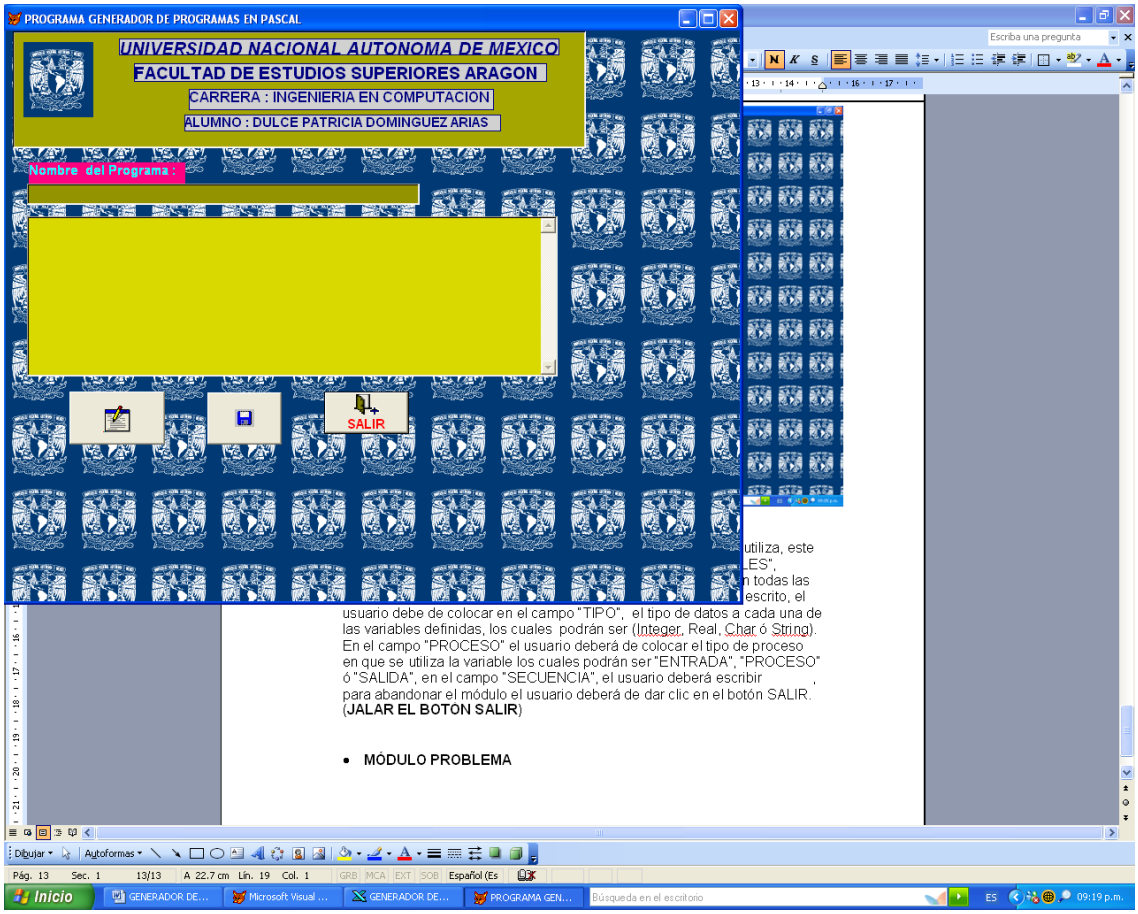
Presenta una tabla que contiene los campos “Texto”, “Paso” y “Orden”, en el campo “Texto”, el usuario escribe las indicaciones pertinentes al Pseudocódigo, en el campo “Paso” , en el campo “Orden” se escribe , el usuario puede escribir tantas líneas como sean necesarias, cada vez que se desee insertar una nueva línea deberá de dar clic en el botón **(jalar el botón de +) ,** la última instrucción del Pseudocódigo deberá ser fin, en el momento que el usuario desee abandonar el módulo deberá dar clic en el botón SALIR **(jalar el botón de SALIR).**

- MÓDULO HERRAMIENTAS



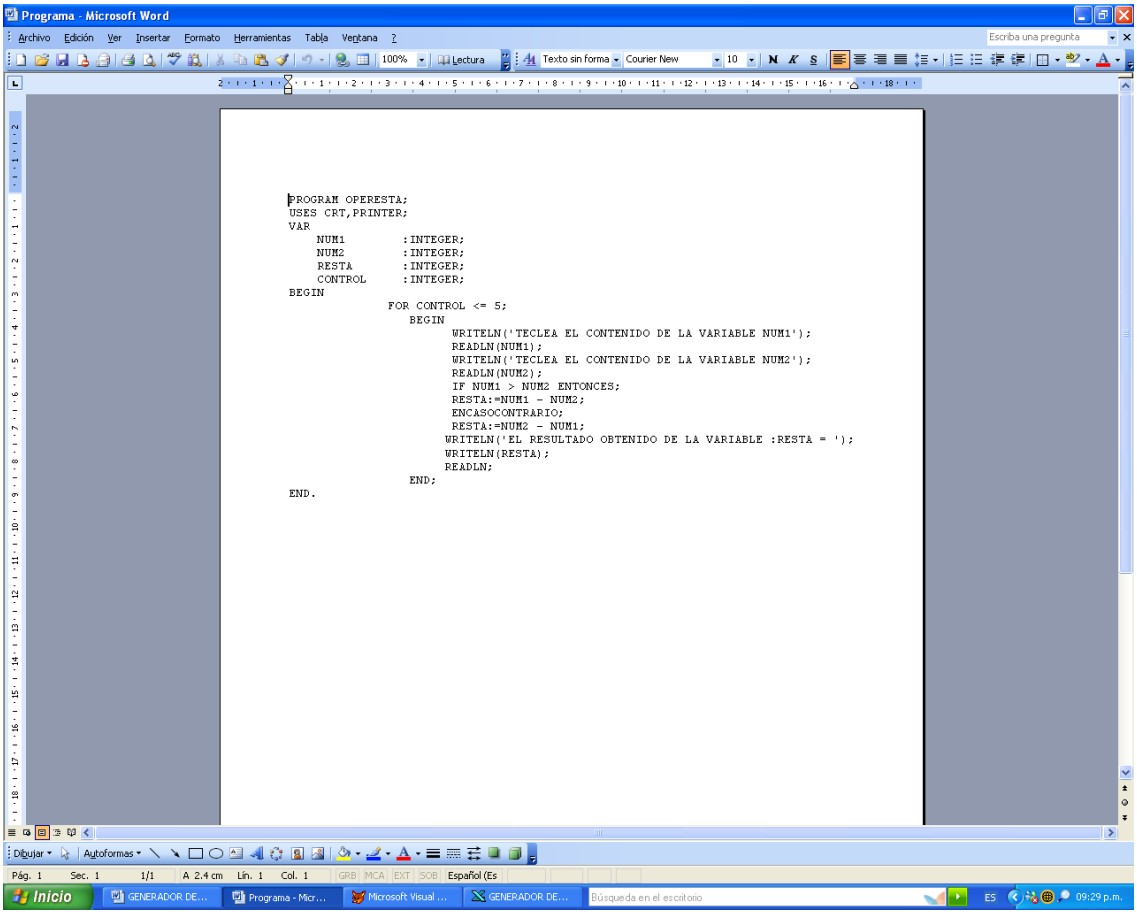
El módulo Herramientas, es el siguiente módulo que el usuario utiliza, este módulo presenta una tabla con los siguientes campos “VARIABLES”, “TIPO”, PROCESO” y “SECUENCIA”, en esta tabla se presentan todas las variables que el Generador reconoció en base al Pseudocódigo escrito, el usuario debe de colocar en el campo “TIPO”, el tipo de datos a cada una de las variables definidas, los cuales podrán ser (Integer, Real, Char ó String). En el campo “PROCESO” el usuario deberá de colocar el tipo de proceso en que se utiliza la variable los cuales podrán ser “ENTRADA”, “PROCESO” ó “SALIDA”, en el campo “SECUENCIA”, el usuario deberá escribir , para abandonar el módulo el usuario deberá de dar clic en el botón SALIR. (**JALAR EL BOTÓN SALIR**)

- MÓDULO PROBLEMA



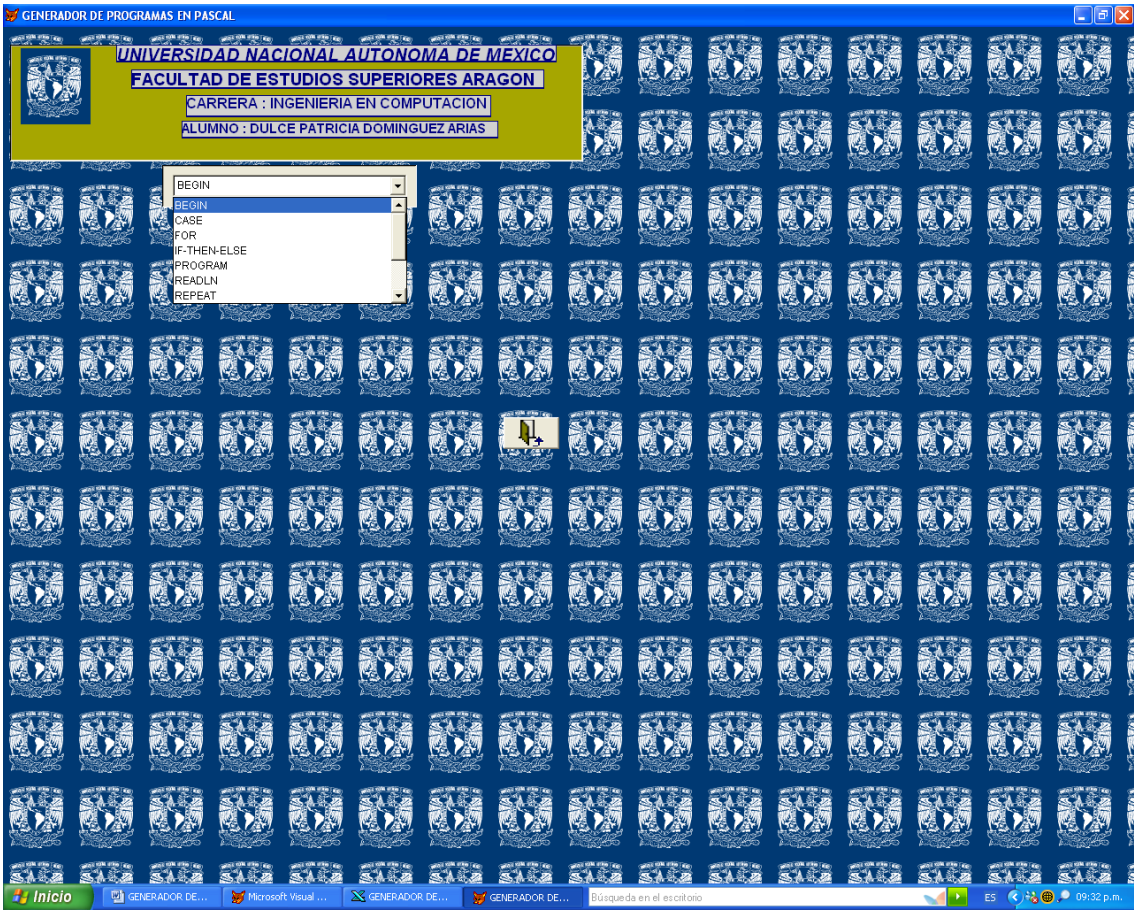
El módulo PROBLEMA, es el siguiente módulo que el usuario utiliza, con la información proporcionada por el usuario, el Generador a construido un programa de cómputo en lenguaje Pascal, el módulo le permite al usuario digitar el nombre del programa, al dar clic en el botón escritura (**jalar el botón escritura**), el programa construido aparece en el área destinada para tal fin (**jalar el área de escritura del programa**), al dar clic en el botón SALVAR (**jalar botón salvar**), el programa se guarda en disco magnético, para abandonar el módulo el usuario deberá dar clic en botón **SALIR**.

- MÓDULO PROCESADOR DE TEXTOS



El modulo Procesador de Textos le permite observar al usuario el programa generado en WORD.

- MÓDULO CONSULTA



El módulo CONSULTA es ayuda en línea que tiene el Generador, con el el usuario puede observar la sintaxis y forma de escritura de los diversos comandos en Pascal.


GENERADOR DE PROGRAMAS EN PASCAL
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGON
CARRERA : INGENIERIA EN COMPUTACION
ALUMNO : DULCE PATRICIA DOMINGUEZ ARIAS
PROGRAM
INSTRUCCION PROGRAM
La instrucción PROGRAM es la primera dentro de un programa en PASCAL, en ella se define el nombre lógico del programa, el formato de la intrucción PROGRAM es.
PROGRAM Nombre del Programa;
Donde el Nombre del Programa podrá ser el que decida el usuario, del tamaño de caracteres que desee con la salvedad de que dicho nombre no deberá tener blancos intermedios.
Ejemplos válidos de la Instrucción PROGRAM son.
Inicio
GENERADOR DE...
Microsoft Visual ...
GENERADOR DE...
GENERADOR DE...
Búsqueda en el escritorio
ES
09:34 p.m.

C a p í t u l o iv

Implantación del sistema

CAPÍTULO IV. IMPLANTACIÓN DEL SISTEMA

IV.1 PRUEBAS GENERALES DEL SISTEMA.

Las pruebas generales del sistema, implican verificar que todos los programas que se interrelacionan en el sistema funcionen de manera correcta. Todos los elementos de entrada son procesados a fin de que se produzcan los resultados previstos, para tener resultados más precisos y confiables.

Antes de la presentación del sistema a los usuarios, se realizaron todas las pruebas pertinentes, en cuanto al funcionamiento y eficiencia del sistema, verificando que la información arrojada, fuera la correcta.

IV.2 PRESENTACIÓN AL USUARIO.

La presentación al usuario del sistema, en este caso a los alumnos del Colegio de Ciencias y Humanidades Plantel Azcapotzalco, se llevó a cabo en el salón de clases, en donde se les dio una introducción de la forma en la que fue creado el sistema y su funcionamiento, además de presentarle los objetivos generales que tendría durante el curso y los beneficios que les brindaría la utilización del sistema.

Se explicó brevemente la manera de acceder al sistema y se desarrolló un ejemplo sencillo con los que habitualmente se comienza la programación en los cursos de Cibernética y Computación I y II.

IV.3 PROGRAMA DE ENSEÑANZA.

Se enseñó a los alumnos de las materias de Cibernética y Computación I y II, como acceder al sistema, elaboraron un ejercicio y se fue explicando paso a paso como funcionaba el sistema y la manera en la que las ventanas iban apareciendo, además de los resultados que se obtendrían, esta clase resultó bastante interesante y amena a los alumnos, los cuales decidieron sin mayor problema utilizar el sistema en clase mientras se estudiaran las instrucciones básicas del Lenguaje de Programación Pascal.

CONCLUSIONES

Lo que puedo concluir es que a lo largo del desarrollo de este proyecto, es que como profesores que laboramos para la Universidad, podemos realizar un sin fin de proyectos en grupos, que puedan ayudar a los alumnos a tener un mejor aprendizaje a lo largo de su paso por el bachillerato, aún cuando las condiciones en la UNAM no sean las más óptimas, la voluntad de un profesor o de varios profesores para engrandecer la docencia en nuestra máxima casa de estudios. La convivencia diaria con los alumnos nos hace crecer como profesores, seres humanos y sobre todo esforzarnos cada día por mejorar nuestras clases, hacer de nuestros jóvenes, personas capaces de crear todo lo que deseen y desarrollarse al final en cualquier ámbito en el que se muevan.

Lo que se creó con este proyecto, fue una herramienta importante para el desarrollo educativo de los alumnos del Colegio de Ciencias y Humanidades del Plantel Azcapotzalco, los propios alumnos probaron el proyecto y además de que les fue fácil su manejo, también lograron al final de las pruebas, comprender los conceptos generales en la elaboración de un programa, comunicarse con el usuario según el problema planteado, visualizar sus resultados de una manera más precisa y dar una mejor presentación a su programa en el editor, ya completando ellos mismos su propio proyecto, resultó ahora que pudieron darse cuenta que su capacidad de razonar y de crear es infinita, siempre y cuando ellos quieran hacerlo.

Se espera que el sistema se instale en una red local en el edificio de cómputo del Colegio de Ciencias y Humanidades Plantel Azcapotzalco, para que pueda ser utilizado por los demás profesores que imparten las asignaturas de Cibernética y Computación I y II, obviamente con el visto bueno de todos, para que los alumnos en general puedan diseñar sus programas con el “Generador”, y tratar de unificar la enseñanza.

También se espera como ya se mencionó con anterioridad en los capítulos de este proyecto, que el alumno sienta que la clase es más amena, que optimice su capacidad de razonamiento y que se interese por la materia, además de que se dará paso seguido la no deserción a dichas asignaturas y por ende un menor índice de reprobación.

Todo lo anterior lo puedo constatar, ya que he puesto en práctica el proyecto con mis propios alumnos y el resultado ha sido bastante satisfactorio.

A P É N D I C E S

PROGRAMAS DE COMPUTADORA

ANALIZA.PRG

```
*SELECT 1
*USE VARIABLE
*ZAP
*INDEX ON VARIABLE TAG VARTAG
*CLOSE DATA
*SELECT 1
*USE PROBLEMA
*SELECT 2
*USE VARIABLE
*SET ORDER TO VARTAG
*SELECT 3
*USE SIGNO
*SET ORDER TO SIGNO
SELECT PROBLEMA
GO TOP
I = 1
VAR      =  SPACE(1)
VARIA    =  0
CAR      =  0
CARACTER =  SPACE(1)
VAR1     =  0
VARARM   =  SPACE(1)
DO WHILE .NOT. EOF()
  IF TEXTO <> "FIN"
    DO WHILE I <= 79
        CARACTER  = SUBSTR(TEXTO,I,1)
       *?"CARACTER"
       *?CARACTER
        *WAIT
        IF SUBSTR(TEXTO,I,1) <> " "
          SELECT SIGNO
          SEEK CARACTER
          IF EOF()
            SELECT PROBLEMA
            CAR   = CAR + 1
            VAR   = VAR + SUBSTR(TEXTO,I,1)
            *?VAR
            *WAIT
          ELSE
              VAR1 = VAL(VAR)
              IF VAR1 > 0
                VAR = SPACE(1)
              ELSE
```

```
                *?VAR
                *WAIT
                IF CAR > 0
                  SELECT VARIABLE
                  VARARM = SUBSTR(VAR,2,CAR)
                  SEEK VARARM
                  IF EOF() .AND. (VARARM <> "ACEPTO")
                      APPEND   BLANK
                   REPLACE  VARIABLE    WITH    SUBSTR(VAR,2,CAR)
                      VAR = SPACE(1)
                      *?"VAR LIMPIA"
                      *?VAR
                      *WAIT
                  ELSE
                       VAR = SPACE(1)
                  ENDIF
                  VARARM = SPACE(1)
                ENDIF
              ENDIF
              SELECT PROBLEMA
              CAR =  0
          ENDIF
        ELSE
          VAR1 = VAL(VAR)
          IF VAR1 > 0
              VAR = SPACE(1)
          ELSE
               SELECT VARIABLE
               VARARM = SUBSTR(VAR,2,CAR)
               SEEK VARARM
               IF EOF() .AND. (VARARM <> "ACEPTO")
                 APPEND BLANK
                 REPLACE  VARIABLE     WITH SUBSTR(VAR,2,CAR)
                 VAR  = SPACE(1)
               ELSE
                  VAR = SPACE(1)
               ENDIF
               VARARM = SPACE(1)
          ENDIF
          SELECT PROBLEMA
          I   = 79
          CAR = 0
        ENDIF
        I = I + 1
    ENDDO
  ENDIF
  I = 1
  SELECT PROBLEMA
  SKIP
  *?"TEXTO"
  *?TEXTO
```

```
   *WAIT
ENDDO
*CLOSE DATA
*SELECT 1
*USE VARIABLE
*GO TOP
*?VARIABLE
*?SUBSTR(VARIABLE,9,1)
*WAIT
*DO WHILE .NOT. EOF()
   *IF VARIABLE = "MOSTRAR" .OR. SUBSTR(VARIABLE,1,1) = """
      *?VARIABLE
      *WAIT
      *DELETE
   *ENDIF
   *SKIP
*ENDDO
*PACK
*CLOSE DATA
SELECT VARIABLE
*USE VARIABLE
SELECT RESERVA
*USE RESERVA
*SET ORDER TO PALABRA
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
   VARAUX   = VARIABLE
   SELECT RESERVA
   SEEK VARAUX
   IF .NOT. EOF()
      SELECT VARIABLE
      DELETE
   ENDIF
   SELECT VARIABLE
   SKIP
ENDDO
SELECT VARIABLE
PACK
RETURN
```

CONSTRUCCION.PRG

```
SELECT PROBLEMA
GO TOP
I     = 1
VAR   = SPACE(1)
CONCAR = 0
CAR   = SPACE(1)
TEXAUX = SPACE(1)
DO WHILE .NOT. EOF()
  DO WHILE I <= 79
    CAR   = SUBSTR(TEXTO,I,1)
    IF CAR <> SPACE(1)
       CONCAR = CONCAR + 1
       VAR   = VAR + SUBSTR(TEXTO,I,1)
    ENDIF
    IF CAR = SPACE(1)
       SELECT RESERVA
       VARAUX = SUBSTR(VAR,2,CONCAR)
       SEEK VARAUX
       IF .NOT. EOF()
         SIGAUX = TRIM(SIGNIFICA)
         SELECT PROBLEMA
         IF VARAUX <> "TERMINAPREGUNTA"
           CONAUX = CONCAR + 1
           DIFAUX = 79 - CONAUX
           TEXAUX = SIGAUX + SUBSTR(TEXTO,CONAUX,DIFAUX)
           REPLACE TEXTO WITH TEXAUX
           IF SIGAUX = "FOR"
              REPLACE PASO WITH "CONTROL"
           ENDIF
         ELSE
            REPLACE TEXTO  WITH SIGAUX
         ENDIF
         I = 79
         CAR   = SPACE(1)
         CONCAR = 0
         VAR   = SPACE(1)
         CONAUX = 0
         DIFAUX = 0
       ELSE
          I    = 79
          VAR  = SPACE(1)
          CAR  = SPACE(1)
          CONCAR= 0
       ENDIF
    ENDIF
    I = I + 1
    SELECT PROBLEMA
  ENDDO
  I = 1
```

```
  SELECT PROBLEMA
  SKIP
ENDDO
DO Form Programa
*THISFORM.ESCRIBE.ENABLED=.T.
*THISFORM.SALVAR.ENABLED=.F.
THISFORM.TEXTO.ENABLED=.T.
THISFORM.CONSTRUCCION.ENABLED=.F.

*DO ANALPRO
*DO ARMAPRO1
```

ARMAPRO1.PRG

```
VAR1   = 0
VAR    = SPACE(1)
*SELECT 2
*USE VARIABLE
*SELECT 3
*USE PROBLEMA
NOMBRE   = SPACE(15)
FILA     = 0
@0,0 CLEAR
@2,10 TO 6,79 DOUBLE
@4,15 SAY "TECLEA NOMBRE DEL PROGRAMA : "
@4,45 GET NOMBRE
READ
SET PRINT ON
SET PRINTER TO FILE PROGRAMA.PRN
SET DEVICE TO PRINT
LINEA1 = "PROGRAM" + SPACE(1) +TRIM(NOMBRE) + ";"
FILA = FILA + 1
@FILA,0     SAY  LINEA1
LINEA1 = "USES CRT,PRINTER;"
FILA = FILA + 1
@FILA,0     SAY LINEA1
LINEA1 = "VAR"
FILA = FILA +  1
@FILA,0     SAY LINEA1
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
  VAR1   =    VAL(VARIABLE)
  VAR    =    SPACE(4) + VARIABLE + ":" + TRIM(TIPO) + ";"
  IF VAR1 = 0 .AND. (PROCESO = "ENTRADA" .OR. PROCESO =
"PROCESO" .OR. PROCESO = "SALIDA")
     FILA = FILA + 1
     @FILA,0   SAY VAR
  ENDIF
  SELECT VARIABLE
  SKIP
ENDDO
VAR = SPACE(4) + "CONTROL" + SPACE(5) + ":" + "INTEGER;"
FILA = FILA + 1
@FILA,0 SAY VAR
LINEA1 = "BEGIN"
FILA   = FILA + 1
@FILA,0  SAY LINEA1
SELECT PROBLEMA
GO TOP
DO WHILE .NOT. EOF()
  IF PASO = "PASO"
     LINEA1    = SPACE(6) + TRIM(TEXTO) + ";"
     FILA   = FILA + 1
```

```
    @FILA,0  SAY LINEA1
    REPLACE PASO WITH "IMPRESO"
  ENDIF
  SELECT PROBLEMA
  SKIP
ENDDO
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
  IF PROCESO = "ENTRADA" .AND. SECUENCIA = "ANTERIOR"
    VAR     = VARIABLE
    LINEA1  = "     WRITELN('TECLEA EL CONTENIDO DE LA VARIABLE " +
TRIM(VAR) + "');"
    FILA    = FILA + 1
    @FILA,0   SAY LINEA1
    LINEA1  = "     READLN("+TRIM(VAR)+");"
    FILA    = FILA + 1
    @FILA,0   SAY LINEA1
    REPLACE SECUENCIA WITH "IMPRESO"
  ENDIF
  SELECT VARIABLE
  SKIP
ENDDO
SELECT PROBLEMA
GO TOP
DO WHILE .NOT. EOF()
  IF PASO = "CONTROL"
    LINEA1    = SPACE(14) + TRIM(TEXTO) + ";"
    FILA   = FILA + 1
    @FILA,0  SAY LINEA1
    IF PASO = "CONTROL"
       LINEA1 = SPACE(17) + "BEGIN"
       FILA   = FILA + 1
       @FILA,0  SAY LINEA1
    ENDIF
    REPLACE PASO WITH "IMPRESO"
  ENDIF
  SELECT PROBLEMA
  SKIP
ENDDO
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
  IF PROCESO = "ENTRADA" .AND. SECUENCIA <> "IMPRESO"
    VAR     = VARIABLE
    LINEA1  = "                   WRITELN('TECLEA EL CONTENIDO DE LA
VARIABLE " + TRIM(VAR) + "');"
    FILA    = FILA + 1
    @FILA,0   SAY LINEA1
    LINEA1  = "                   READLN("+TRIM(VAR)+");"
    FILA    = FILA + 1
```

```
    @FILA,0   SAY LINEA1
  ENDIF
  SELECT VARIABLE
  SKIP
ENDDO
SELECT PROBLEMA
GO TOP
DO WHILE .NOT. EOF()
  LINEA1    = SPACE(23) + TRIM(TEXTO) + ";"
  VAR1      = VAL(TEXTO)
  IF TEXTO <> "FIN" .AND. VAR1 = 0 .AND. SUBSTR(TEXTO,1,6) <>
"ACEPTO" .AND. PASO <> "IMPRESO"
    FILA   = FILA + 1
    @FILA,0  SAY LINEA1
  ENDIF
  SELECT PROBLEMA
  SKIP
ENDDO
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
  IF PROCESO = "SALIDA"
    VAR     = VARIABLE
    LINEA1  = "                  WRITELN('EL RESULTADO OBTENIDO DE LA
VARIABLE :" + TRIM(VAR) + " = " + "');"
    FILA    = FILA + 1
    @FILA,0   SAY LINEA1
    LINEA1  = "                  WRITELN("+TRIM(VAR)+");"
    FILA    = FILA + 1
    @FILA,0   SAY LINEA1
  ENDIF
  SELECT VARIABLE
  SKIP
ENDDO
LINEA1 = "                 READLN;"
FILA   = FILA + 1
@FILA,0  SAY  LINEA1
LINEA1 = "               END;"
FILA   = FILA + 1
@FILA,0  SAY  LINEA1
LINEA1 ="END."
FILA   = FILA + 1
@FILA,0  SAY LINEA1
SET PRINT OFF
SET DEVICE TO SCREEN
CLOSE DATA
RETURN
ESCRIBE.PRG

VAR1   = 0
VAR    = SPACE(1)
```

```
FILA    = 0
LINEA1 = ''
LINEA1 = LINEA1 + "PROGRAM" + SPACE(1)
+ALLTRIM(THISFORM.NOMBRE.VALUE) + ";" + CHR (13)
LINEA1 = LINEA1 + "USES CRT,PRINTER;" + CHR(13)
LINEA1 = LINEA1 + "VAR" + CHR(13)
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
  VAR1   =   VAL(VARIABLE)
  VAR    =   SPACE(4) + VARIABLE + ":" + TRIM(TIPO) + ";"
  IF VAR1 = 0 .AND. (PROCESO = "ENTRADA" .OR. PROCESO =
"PROCESO" .OR. PROCESO = "SALIDA")
    LINEA1 = LINEA1 + VAR + CHR(13)
  ENDIF
  SELECT VARIABLE
  SKIP
ENDDO
VAR = SPACE(4) + "CONTROL" + SPACE(5) + ":" + "INTEGER;"
LINEA1 = LINEA1 + VAR + CHR(13)
VAR = "BEGIN"
LINEA1 = LINEA1 + VAR + CHR(13)
NUM = 0
SELECT PROBLEMA
GO TOP
NUM = NUM + 1
DO WHILE .NOT. EOF()
  IF PASO = "PASO" .AND. ORDEN = 1
    VAR    = SPACE(6) + TRIM(TEXTO) + ";"
    LINEA1 = LINEA1 + VAR + CHR(13)
    REPLACE PASO WITH "IMPRESO"
  ENDIF
  SELECT PROBLEMA
  SKIP
ENDDO
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
  IF PROCESO = "ENTRADA" .AND. SECUENCIA = "ANTERIOR"
    VAR     = VARIABLE
    LINEA1  = LINEA1 + "      WRITELN('TECLEA EL CONTENIDO DE LA
VARIABLE " + TRIM(VAR) + "');" + CHR(13)
    LINEA1  = LINEA1 + "      READLN("+TRIM(VAR)+");" + CHR(13)
    REPLACE SECUENCIA WITH "IMPRESO"
  ENDIF
  SELECT VARIABLE
  SKIP
ENDDO
SELECT PROBLEMA
GO TOP
NUM = NUM + 1
```

```
DO WHILE .NOT. EOF()
  IF PASO = "PASO" .AND. ORDEN = 2
    VAR    = SPACE(6) + TRIM(TEXTO) + ";"
    LINEA1 = LINEA1 + VAR + CHR(13)
    REPLACE PASO WITH "IMPRESO"
  ENDIF
  SELECT PROBLEMA
  SKIP
ENDDO
SELECT PROBLEMA
GO TOP
DO WHILE .NOT. EOF()
  IF PASO = "CONTROL"
    LINEA1    = LINEA1 + SPACE(14) + TRIM(TEXTO) + ";" + CHR(13)
    IF PASO = "CONTROL"
      LINEA1 = LINEA1 + SPACE(17) + "BEGIN" + CHR(13)
    ENDIF
    REPLACE PASO WITH "IMPRESO"
  ENDIF
  SELECT PROBLEMA
  SKIP
ENDDO
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
  IF PROCESO = "ENTRADA" .AND. SECUENCIA <> "IMPRESO"
    VAR     = VARIABLE
    LINEA1  = LINEA1 + "                    WRITELN('TECLEA EL CONTENIDO
DE LA VARIABLE " + TRIM(VAR) + "');" + CHR(13)
    LINEA1  = LINEA1 + "                    READLN("+TRIM(VAR)+");" + CHR(13)
  ENDIF
  SELECT VARIABLE
  SKIP
ENDDO
VUEL = 0
SELECT PROBLEMA
GO TOP
DO WHILE .NOT. EOF()
  VAR    = SPACE(23) + TRIM(TEXTO) + ";"
  VAR1   = VAL(TEXTO)
  IF TEXTO <> "FIN" .AND. VAR1 = 0 .AND. SUBSTR(TEXTO,1,6) <>
"ACEPTO" .AND. PASO <> "IMPRESO"
    LINEA1    = LINEA1 + VAR + CHR(13)
  ENDIF
  IF TEXTO = "FINREPITE" .AND. ORDEN = 1
    VAR= "END;"
    LINEA1=LINEA1+ VAR + CHR(13)
    VUEL = 1
  ENDIF
  SELECT PROBLEMA
  SKIP
```

```
ENDDO
SELECT VARIABLE
GO TOP
DO WHILE .NOT. EOF()
  IF PROCESO = "SALIDA"
    VAR     = VARIABLE
    LINEA1  = LINEA1 + "                  WRITELN('EL RESULTADO
OBTENIDO DE LA VARIABLE :" + TRIM(VAR) + " = " + "');" + CHR(13)
    LINEA1  = LINEA1 + "                  WRITELN("+TRIM(VAR)+");" +
CHR(13)
  ENDIF
  SELECT VARIABLE
  SKIP
ENDDO
LINEA1 = LINEA1 + "                   READLN;" + CHR(13)
IF VUEL = 0
  LINEA1 = LINEA1 + "               END;" + CHR(13)
ENDIF
LINEA1 = LINEA1 + "END." + CHR(13)
ThisForm.mDespliega.Value = Linea1
Select Pascal
Go Top
Replace Pascal.Programa With Linea1
```

PROGRAMA: ANALIZA.PRG

Objetivo:

Parte 1. Analizar el contenido de la base de datos Problema, separando variables de caracteres especiales, para posteriormente actualizar la base de datos Variable.

Parte 2. Analizar el contenido de la base de datos Variable con la finalidad de borrar los registros que sean palabras reservadas.

Entrada: Base de datos **Problema**.
Base de datos **Signo** indexado por el campo signo
Base de datos **Reserva** indexado por el campo palabra

Salida: Base de datos **Variable** indexado por el campo variable

Procedimiento:

Parte 1

Selecciona la base de datos Problema y coloca el apuntador de registros al inicio de la base de datos.
Inicializa las variables de control.
Inicio ciclo para recorrer la base de datos Problema mientras no sea fin de archivo
Pregunta Si el campo TEXTO de la base datos PROBLEMA <> “FIN”
Inicializa variable de control I con 1
Inicia CICLO para recorrer carácter a carácter el campo TEXTO
Si carácter seleccionado es <> de 1 espacio
Selecciona la base de datos Signo
Realiza búsqueda del carácter seleccionado
Si la bandera de búsqueda es fin de archivo
Incrementa en 1 el contador de caracteres
Guarda carácter satisfactorio a la variable VAR
Si la bandera de búsqueda no es fin de archivo
Es un carácter que esta definido en la Base de datos Signo
Lo desecha
En caso contrario el carácter seleccionado es espacio
Incrementa 1 al contador de espacios.
Si el contador de espacios es igual a 2
Indica que el contenido del campo TEXTO se ha terminado
Coloca en la variable VARARM el contenido de la variable VAR
Selecciona la base de datos Variable
Realiza búsqueda con contenido de la variable VARARM
Si la bandera de búsqueda es fin de archivo
Adiciona un registro en blanco en la base de datos VARIABLE
Actualiza el campo Variable con contenido de VARARM
Si la bandera de búsqueda no es fin de archivo
Indica que el contenido de la variable VARARM ya fue
Se encuentra en la base de datos Variable
En caso contrario el contador de espacios es igual a 1
Indica que debe seguir recorriendo el campo TEXTO.
Adiciona 1 a la variable de control I.

Fin de ciclo para recorrer campo TEXTO
Selecciona la base de datos PROBLEMA
Salta al siguiente registro
En caso contrario
Desecha el registro el cual el contenido del campo TEXTO = “FIN”
Fin de Ciclo para recorrer la base de datos Problema.
Cierra las bases de datos

Parte 2

Selecciono la base de datos Variable y coloco el apuntador de registros al inicio de la base de datos
Inicio ciclo para recorrer la base de datos Variable mientras no sea fin de archivo
Asigno a la variable VARAUX con el contenido del campo VARIABLE
Selecciona la base de datos Reserva
Realiza búsqueda del campo seleccionado
Si la bandera de búsqueda no es fin de archivo
Selecciona la base de datos Variable
Marca el registro para ser borrado
Selecciona la base de datos variable
Salta al siguiente registro
Fin de Ciclo para recorrer la base de datos Variable
Selecciono la base de datos Variable
Borro los registros marcados a ser dados de baja

PROGRAMA: CONSTRUCCION.PRG

Objetivo: Analizar el contenido de la base de datos Problema, con la finalidad de modificar la escritura del Pseudocódigo a lenguaje Pascal.

Entrada: Base de datos **Problema**.
Base de datos **Reserva** indexado por el campo palabra

Salida: Base de datos Problema indexado por el campo variable

Procedimiento:

Selecciona la base de datos Problema y coloca el apuntador de registros al inicio de la base de datos.
Inicializa las variables de control.
Inicio ciclo para recorrer la base de datos Problema mientras no sea fin de archivo
    Inicializa variable de control I con 1
    Inicia CICLO para recorrer carácter a carácter el campo TEXTO
      Si carácter seleccionado es <> de 1 espacio
        Incrementa en 1 el contador de caracteres
        Guarda carácter satisfactorio a la variable VAR
      Fin pregunta
      Si carácter seleccionado = 1 espacio
        Asigno a la variable VARAUX con el contenido del campo VAR
        Selecciona la base de datos Reserva
        Realiza búsqueda del campo seleccionado
        Si la bandera de búsqueda no es fin de archivo
          Asigno la variable Sigaux con el contenido de la variable Significa
          Selecciona la base de datos Problema
          Asigno en la variable Texaux contenido de Sigaux + Texto
          Actualizo el contenido de la variable Texto con Texaux
        Encaso Contrario
        Termino Ciclo
        Inicializo contadores
      Encaso contrario
      Incremento en 1 a la variable de control I
    Fin ciclo
    Selecciona la base de datos Problema
    Salta al registro siguiente
Fin ciclo

PROGRAMA: ARMAPRO1.PRG

Objetivo: Construir el programa en lenguaje de programación PASCAL empleando para tal fin las bases de datos PROBLEMA y VARIABLE.

Entrada: Base de datos **Problema**.

Base de datos **Variable**

Salida: Archivo de Impresión **Programa.Prn**

Procedimiento:

Se le solicita al usuario que digite a través del teclado el nombre del programa, se recomienda que el nombre seleccionado sea acorde al problema resuelto.

Se construye la parte relativa al encabezado de un programa en PASCAL, es decir se ensambla las palabras.

PROGRAM Nombre del Programa;
USES Crt,Printer;
Var

Se selecciona la Base de Datos **Variable** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Variable**.
Se almacena en la variable **Var** el nombre de la variable y su tipo de datos.
Si el contenido de la variable Proceso de la base de datos **Variable** es igual a “Entrada” “Proceso” o “Salida”
adiciona una línea con el nombre de la variable y su tipo de dato al archivo de impresión Programa.Prn.
Fin pregunta
Selecciono la base de datos **Variable**
Salto al registro siguiente
Fin ciclo
Se almacena la variable **Var** con la palabra Control tipo de dato Entero, que será la variable de control que se asociará a la instrucción For en el caso de que el programa construido maneje ciclos.
Se construye la parte relativa al inicio del cuerpo de un programa en PASCAL es decir se ensambla la palabra.

Begin

Se selecciona la Base de Datos **Problema** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Problema**.
Si el contenido de la variable Paso de la base de datos **Problema** es igual Paso
Se almacena en la variable Línea el contenido de la variable Texto de la base de datos **Problema**
Se adiciona una línea al archivo de impresión Programa.Prn
Se actualiza el campo Paso de la base de datos **Problema** con la palabra “IMPRESO”
Fin pregunta
Selecciono la base de datos **Problema**

Salto al registro siguiente
Fin ciclo
Se selecciona la Base de Datos **Variable** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Variable**.
Si el contenido de las variables Proceso y Secuencia de la base de datos **Variable** es igual a “ENTRADA” y “ANTERIOR” respectivamente
Se adiciona una línea al archivo de impresión Programa.Prn con la instrucción

WRITELN(“Teclea el contenido de la variable” + VAR);

Se adiciona una línea al archivo de impresión Programa.Prn con la instrucción

READLN(VAR);

Se actualiza el campo Secuencia de la base de datos **Variable** con la palabra “IMPRESO”
Fin pregunta
Selecciono la base de datos **Variable**
Salto al registro siguiente
Fin ciclo
Se selecciona la Base de Datos **Problema** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Problema**.
Inicializo la variable Línea1 con el contenido de la variable TEXTO de la base de datos **Problema**
Si TEXTO diferente “FIN” “ACEPTO” y “IMPRESO”
Se adiciona una línea al archivo de impresión Programa.Prn
Fin pregunta
Selecciono la base de datos **Problema**
Salto al registro siguiente
Fin de ciclo
Se selecciona la Base de Datos **Variable** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Variable**.
Si el contenido de las variable Proceso de la base de datos **Variable** es igual a “SALIDA”
Asigno a la variable VAR con el contenido de Variable de la base de datos **Variable**
Se adiciona una línea al archivo de impresión Programa.Prn con la instrucción

WRITELN(“El resultado obtenido de la variable “ + VAR);
Fin pregunta
Selecciono la base de datos **Variable**
Salto al registro siguiente

Se construye la parte relativa al final de un programa en PASCAL, es decir se ensambla las palabras.

```
            READLN;
        END;
END.
```

PROGRAMA : ESCRIBE.PRG

Objetivo: Construir el programa en lenguaje de programación PASCAL enviándolo a pantalla con la finalidad de que el usuario observe en un menú desplegable el programa construido, empleando para ello las bases de datos PROBLEMA y VARIABLE.

Entrada: Base de datos **Problema**.
Base de datos **Variable**

Salida: Menú Desplegable.

Procedimiento:

Se le solicita al usuario que digite a través del teclado el nombre del programa, se recomienda que el nombre seleccionado sea acorde al problema resuelto.

Se construye la parte relativa al encabezado de un programa en PASCAL, es decir se ensambla las palabras.

Se emplea la variable Línea1 en la cual se almacenaran las instrucciones que se vayan generando, la secuencia **CHR(13)** indica salto de línea.
Se inicializa la variable Línea1 con un blanco.

Se almacena en la variable Linea1 la palabra **PROGRAM** Nombre del programa + **CHR(13)**
Se almacena en la variable Línea1 la palabra **USES** Crt,Printer + **CHR(13)**
Se almacena en la variable Línea1 la palabra **Var** + **CHR(13)**

Se selecciona la Base de Datos **Variable** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Variable**.
    Se almacena en la variable **Var** el nombre de la variable y su tipo de datos.
     Si el contenido de la variable Proceso de la base de datos **Variable** es igual a “Entrada” “Proceso” o “Salida”
            adiciona a la variable Línea1 con el nombre de la variable y su tipo de dato + **CHR(13)**.
     Fin pregunta
    Selecciono la base de datos **Variable**
    Salto al registro siguiente
Fin ciclo
Se almacena la variable **Var** con la palabra Control tipo de dato Entero, que será la variable de control que se asociará a la instrucción For en el caso de que el programa construido maneje ciclos.
Se construye la parte relativa al inicio del cuerpo de un programa en PASCAL es decir se ensambla la palabra.

Se almacena en la variable Línea1 la palabra **Begin** + **CHR(13)**

Se selecciona la Base de Datos **Problema** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Problema**.
    Si el contenido de la variable Paso de la base de datos **Problema** es igual Paso
            Se almacena en la variable Línea1 el contenido de la variable Texto de la base de datos **Problema** + **CHR(13)**
            Se actualiza el campo Paso de la base de datos **Problema** con la palabra “IMPRESO”
    Fin pregunta
    Selecciono la base de datos **Problema**

Salto al registro siguiente
Fin ciclo
Se selecciona la Base de Datos **Variable** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Variable**.
Si el contenido de las variables Proceso y Secuencia de la base de datos **Variable** es igual a “ENTRADA” y “ANTERIOR” respectivamente
Se adiciona en la variable Linea1 con la instrucción WRITELN(“Teclea el contenido de la variable” + VAR) + **CHR(13)**;
Se adiciona en la variable Línea1 con la instrucción READLN(VAR) + **CHR(13)**;
Se actualiza el campo Secuencia de la base de datos **Variable** con la palabra “IMPRESO”
Fin pregunta
Selecciono la base de datos **Variable**
Salto al registro siguiente
Fin ciclo
Se selecciona la Base de Datos **Problema** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Problema**.
Inicializo la variable Línea1 con el contenido de la variable TEXTO de la base de datos **Problema**
Si TEXTO diferente “FIN” “ACEPTO” y “IMPRESO”
Se adiciona en la variable Línea1 la variable VAR + **CHR(13)**
Fin pregunta
Selecciono la base de datos **Problema**
Salto al registro siguiente
Fin de ciclo
Se selecciona la Base de Datos **Variable** y se coloca el apuntador al inicio de la base de datos.
Inicio ciclo para recorrer la base de datos **Variable**.
Si el contenido de las variable Proceso de la base de datos **Variable** es igual a “SALIDA”
Asigno a la variable VAR con el contenido de Variable de la base de datos **Variable**
Se adiciona en la variable Línea1 con la instrucción WRITELN(“El resultado obtenido de la variable “ + VAR); + **CHR(13)**
Fin pregunta
Selecciono la base de datos **Variable**
Salto al registro siguiente

Se construye la parte relativa al final de un programa en PASCAL, es decir se ensambla las palabras.

Se adiciona a la variable Línea1 la instrucción READLN; + **CHR(13)**

Se adiciona a la variable Línea1 la instrucción END; + **CHR(13)**

Se adiciona a la variable Línea1 la instrucción **END.** + **CHR(13)**

Se ejecuta el comando ThisForm.mDespliega.Value=Línea1

Se selecciona la base de datos Pascal
Se coloca el apuntador al inicio de la base de datos
Se actualiza la base de datos Pascal.programa con Línea1.

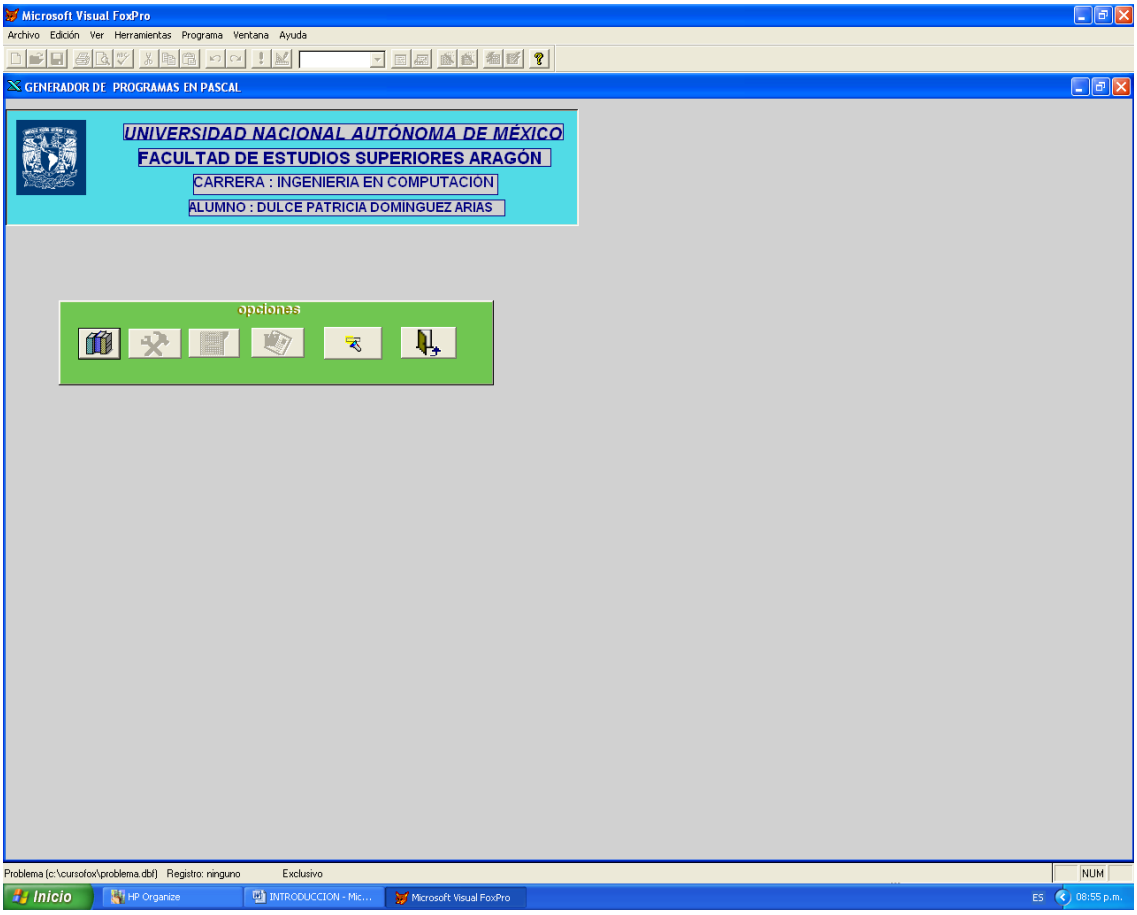
MANUAL DE USUARIO
DEL "GENERADOR DE PROGRAMAS EN PASCAL"

Para poder utilizar el Generador de programas en pascal, basta con seguir el siguiente ejemplo.

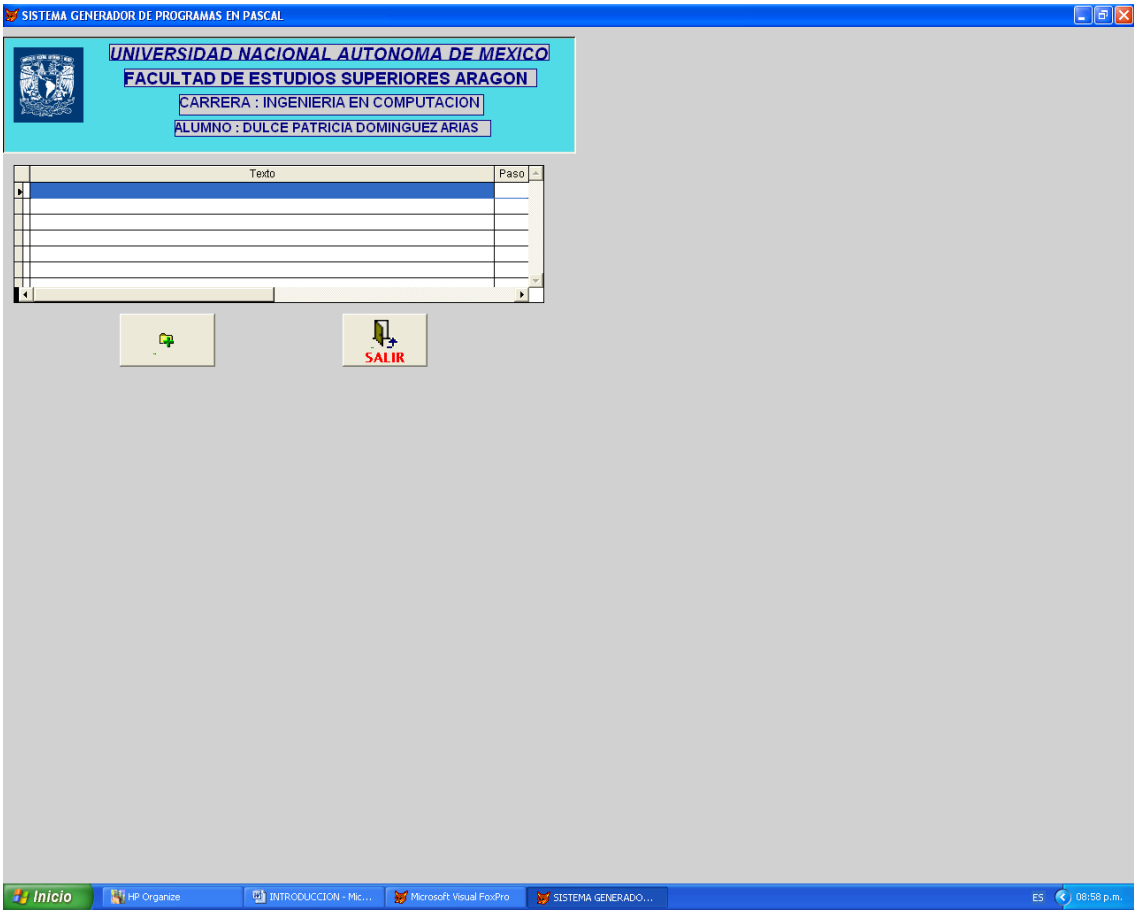
Se desea calcular la fuerza aplicada a un cuerpo, sabiendo que FUERZA=MASA*ACELERACION, los datos de masa y aceleración, deberán ser proporcionados por el usuaro.

1.-

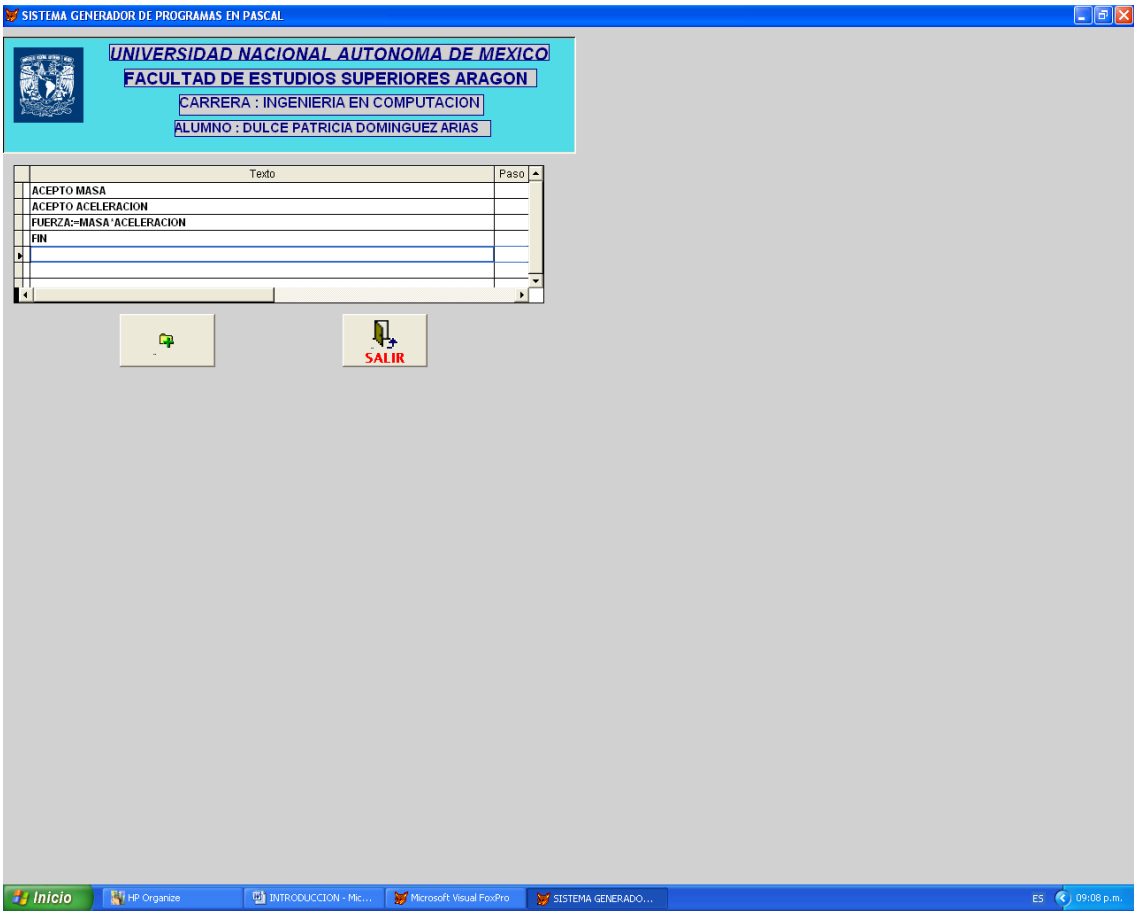
2.-



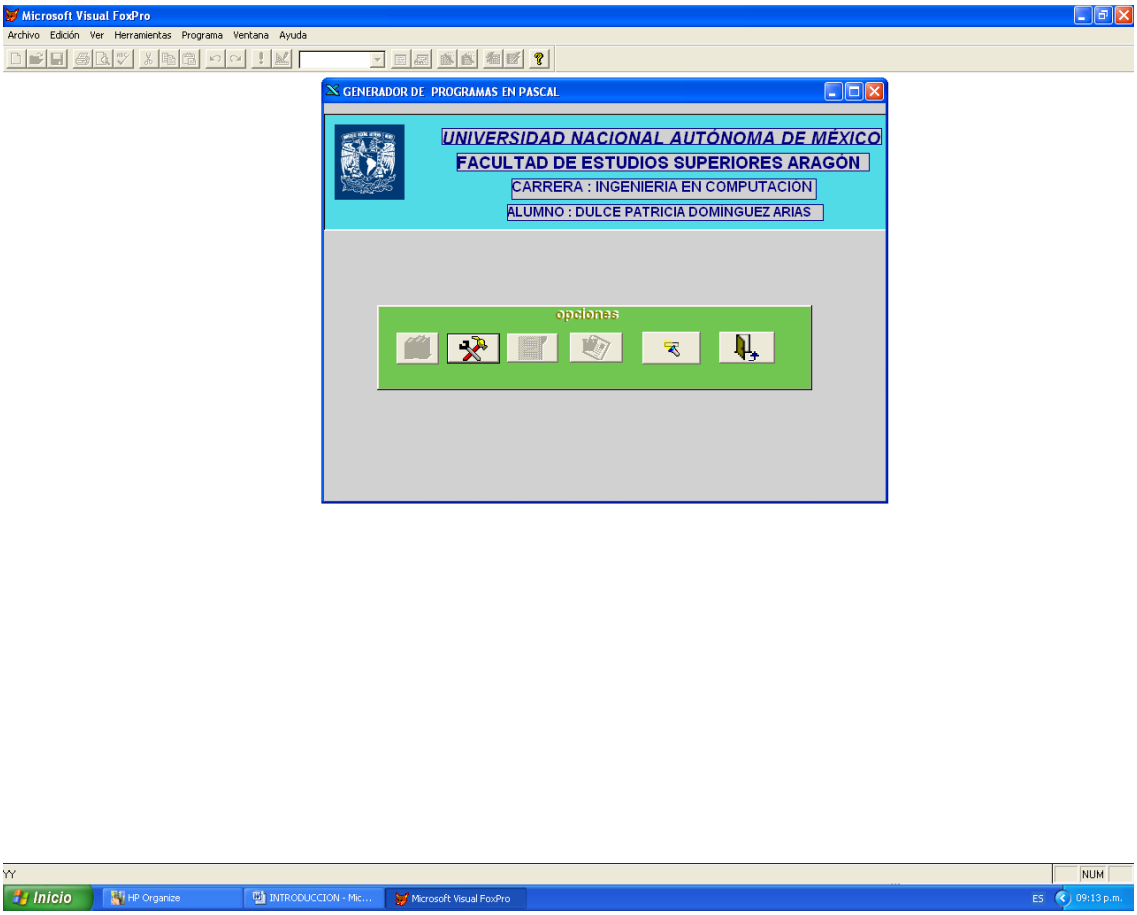
Enseguida da un clic en el ícono que tiene por imagen algunos libros



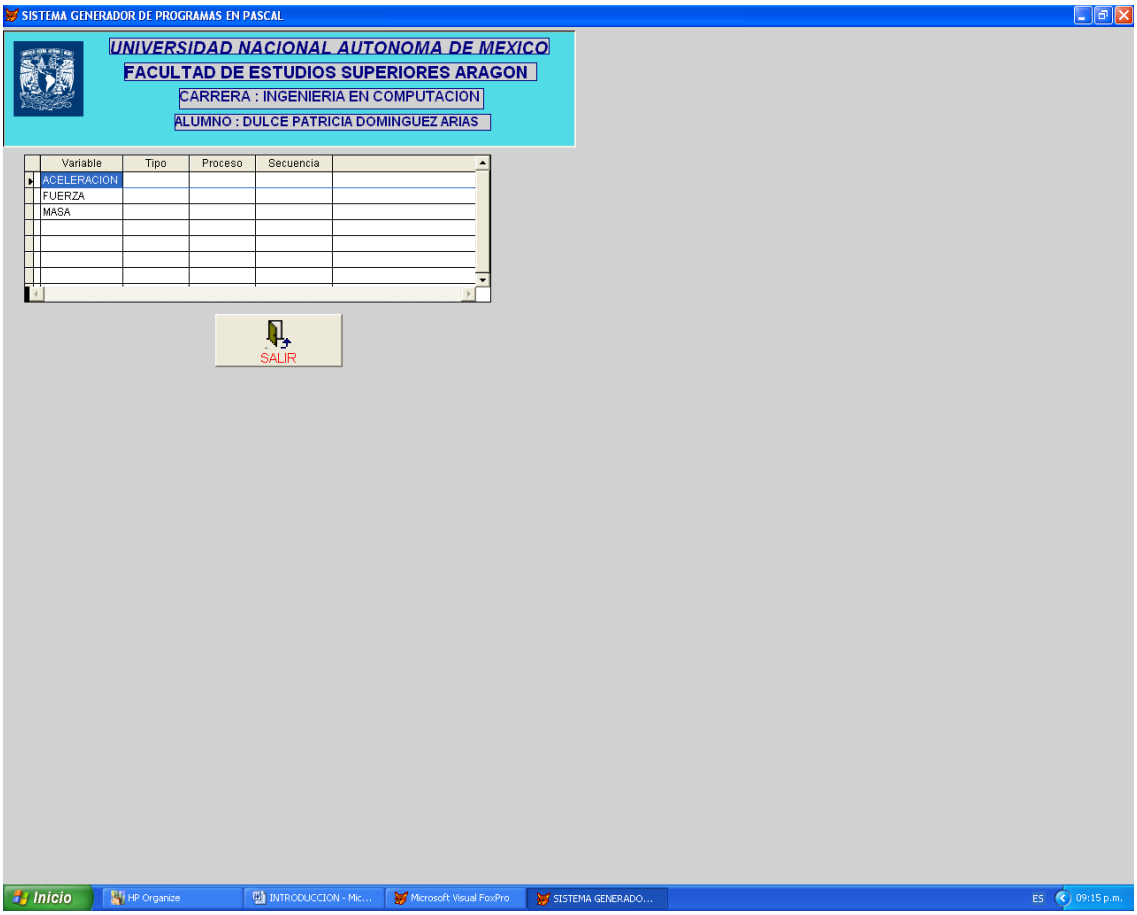
Podremos ahora capturar el algoritmo de solución del problema dando clic en el ícono que tiene dibujado el signo **+** cada vez que desees insertar una línea, cuando el problema requiera la introducción de datos, deberás teclear la palabra ACEPTO (con mayúsculas), seguida de la variable que guardará la información.



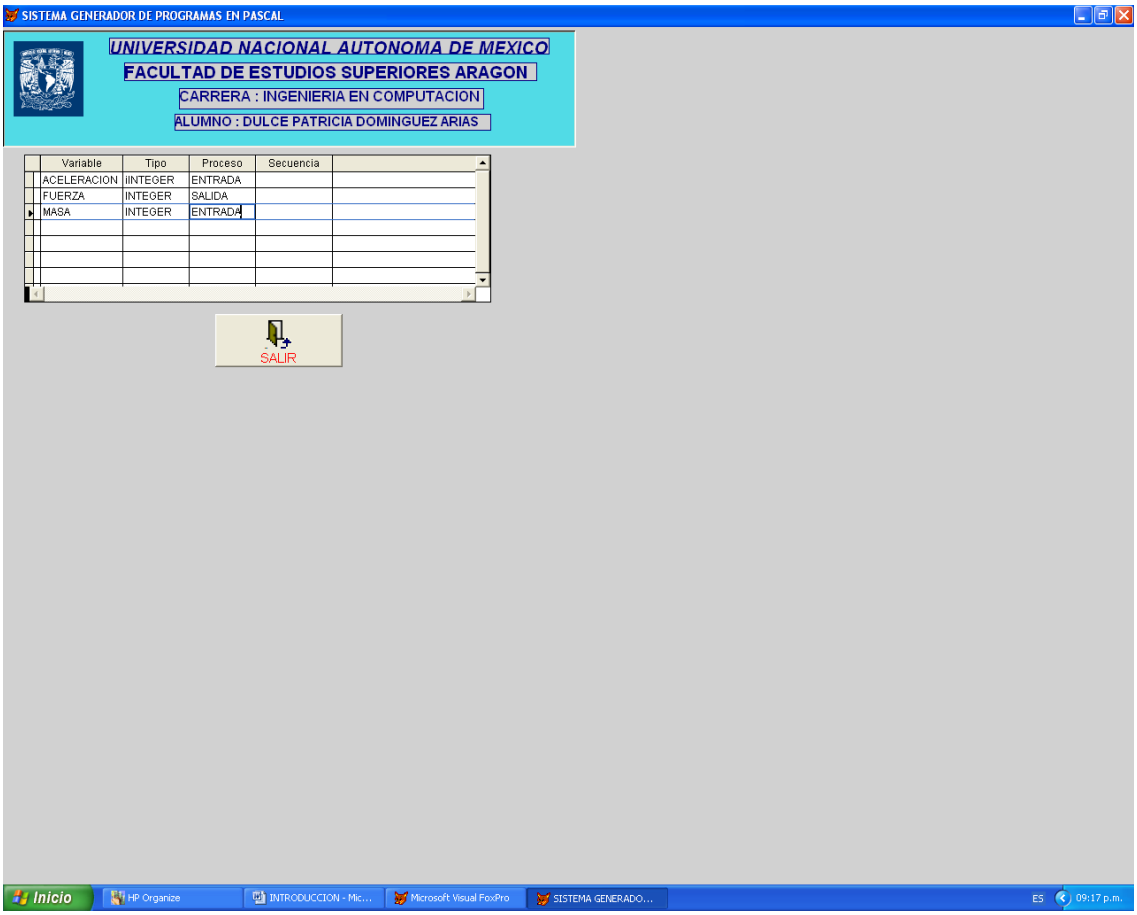
Enseguida, da un clic en el botón salir, para continuar con el siguiente paso.



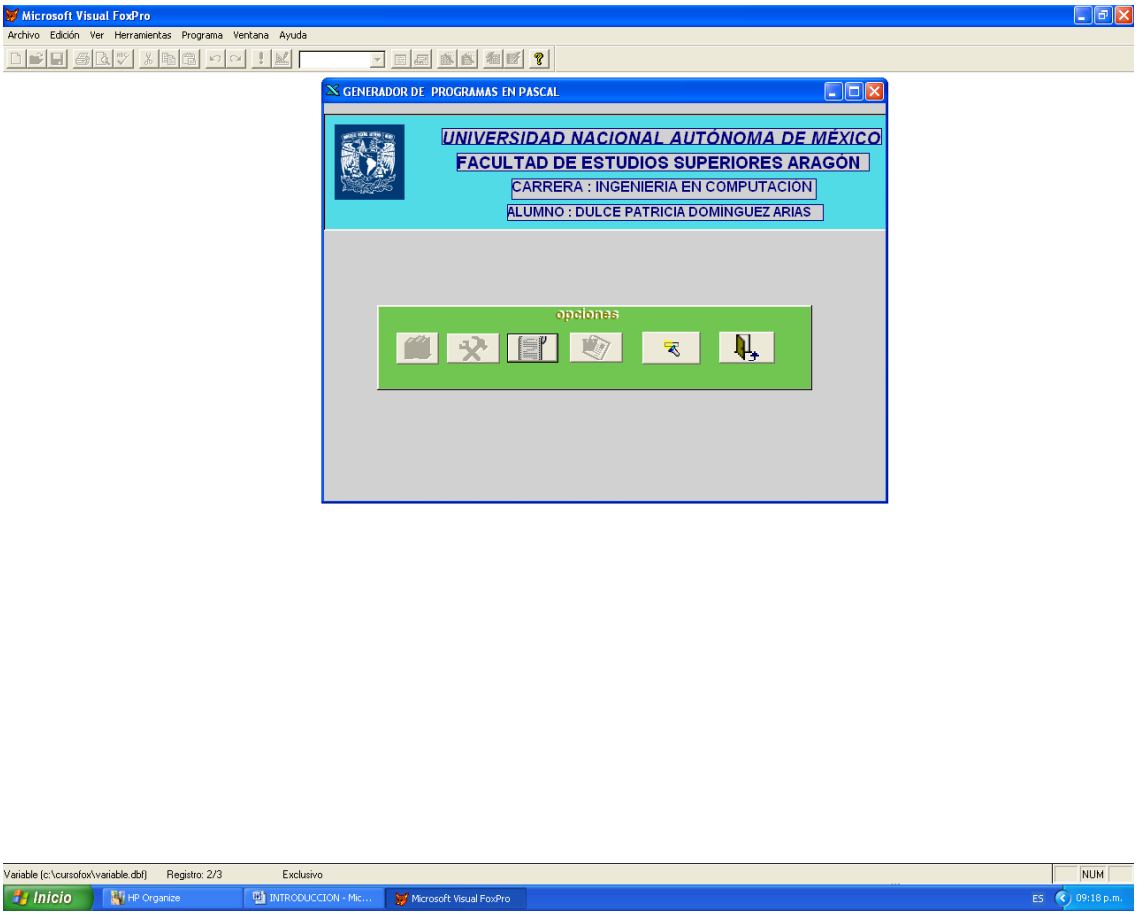
En el botón de las herramientas, podrás ahora definir las variables que usas dentro del algoritmo, recuerda que existen variables de entrada, proceso y salida.



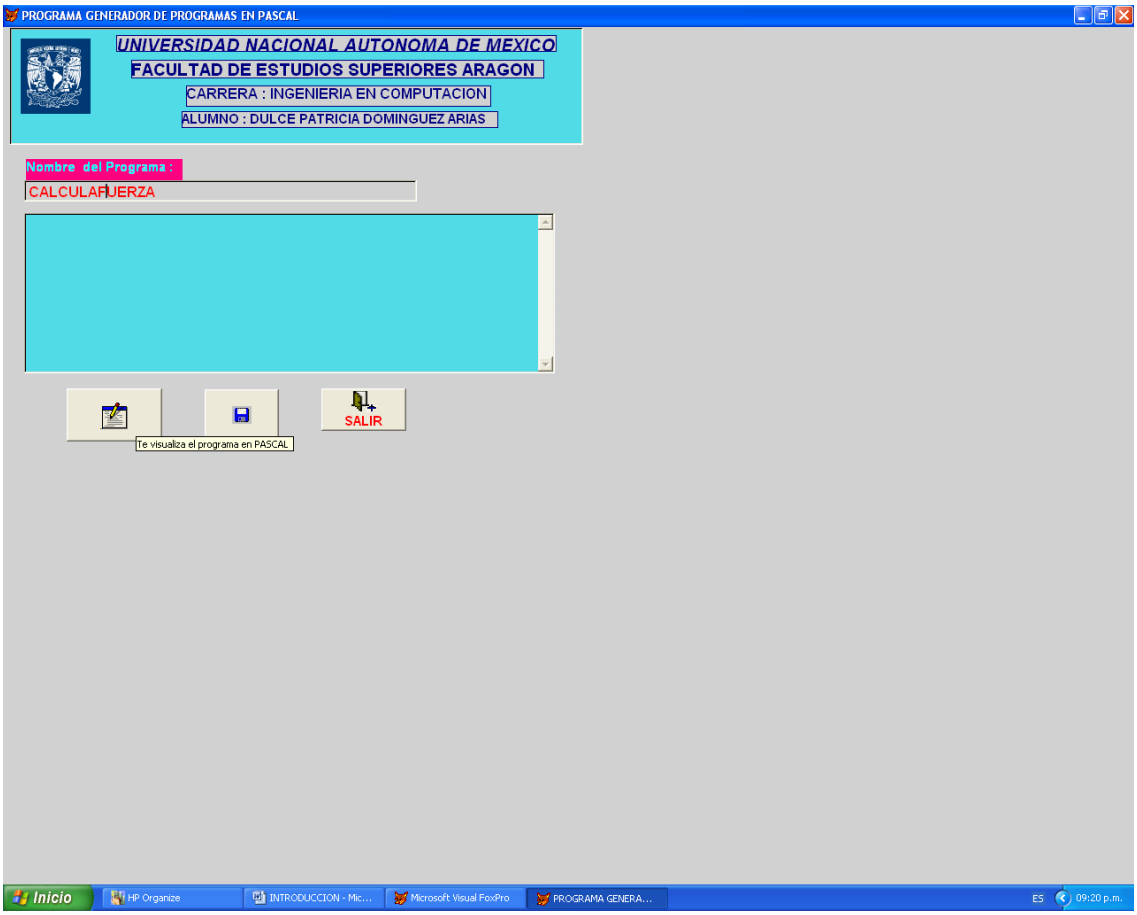
El programa ha generado ya las variables que se utilizaron en el paso anterior, ahora solo habrá que indicarle si son integer, real, string o char y si son de entrada, de proceso o de salida.



Deberás ahora dar un clic en el botón salir y continuar en el siguiente paso.

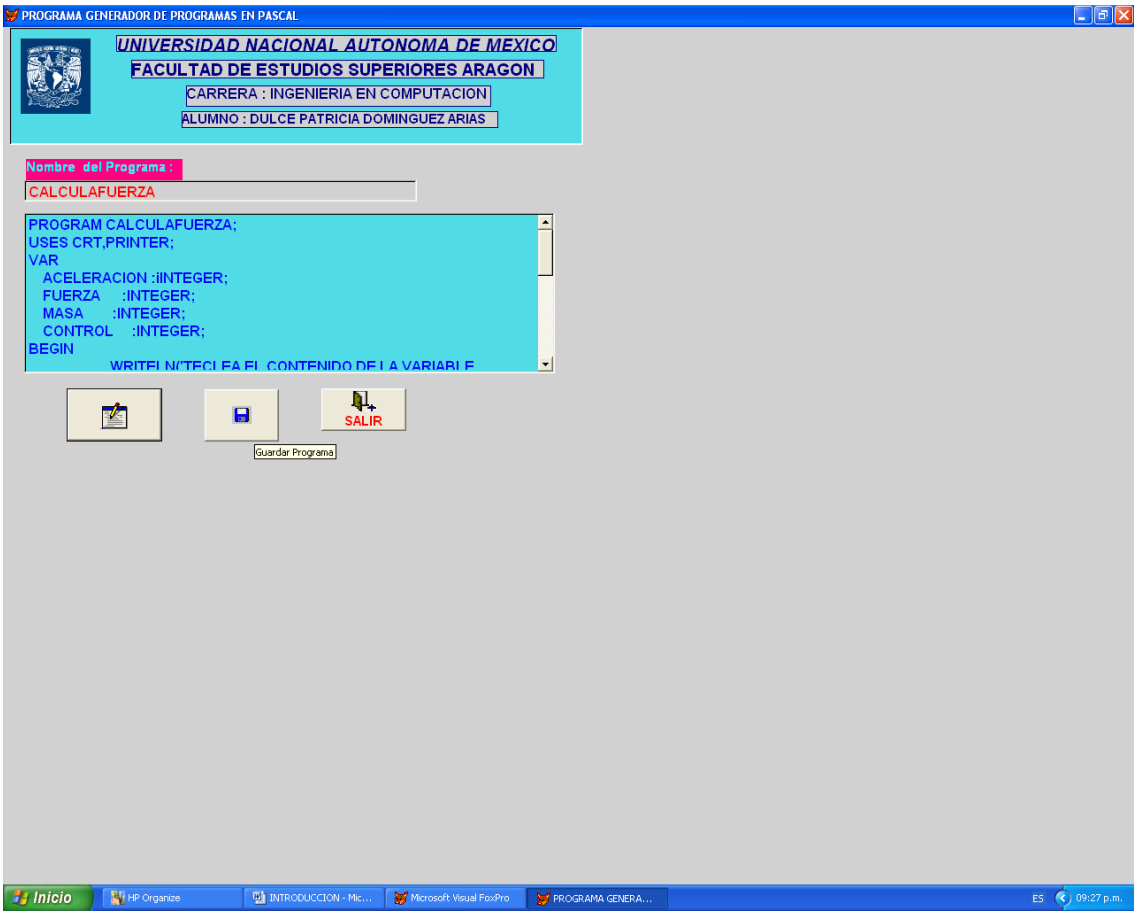


En esta opción nos lleva al módulo de construcción del programa.



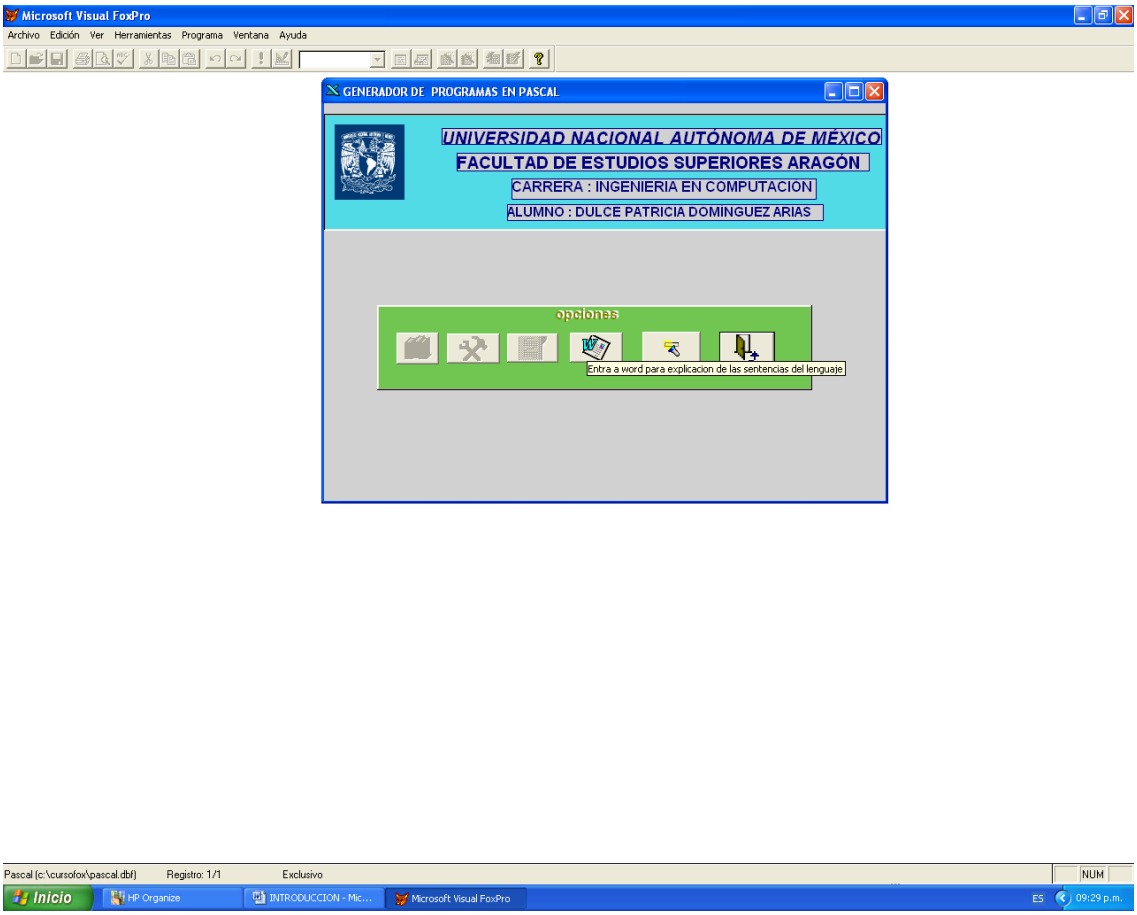
Deberás indicar el nombre del programa, con las reglas que ya conoces, no usar el mismo nombre de alguna variable, y no escribir espacios.

En el botón que tiene la hoja con el lápiz, podrás dar un clic para que genere de manera automática el código en pascal.

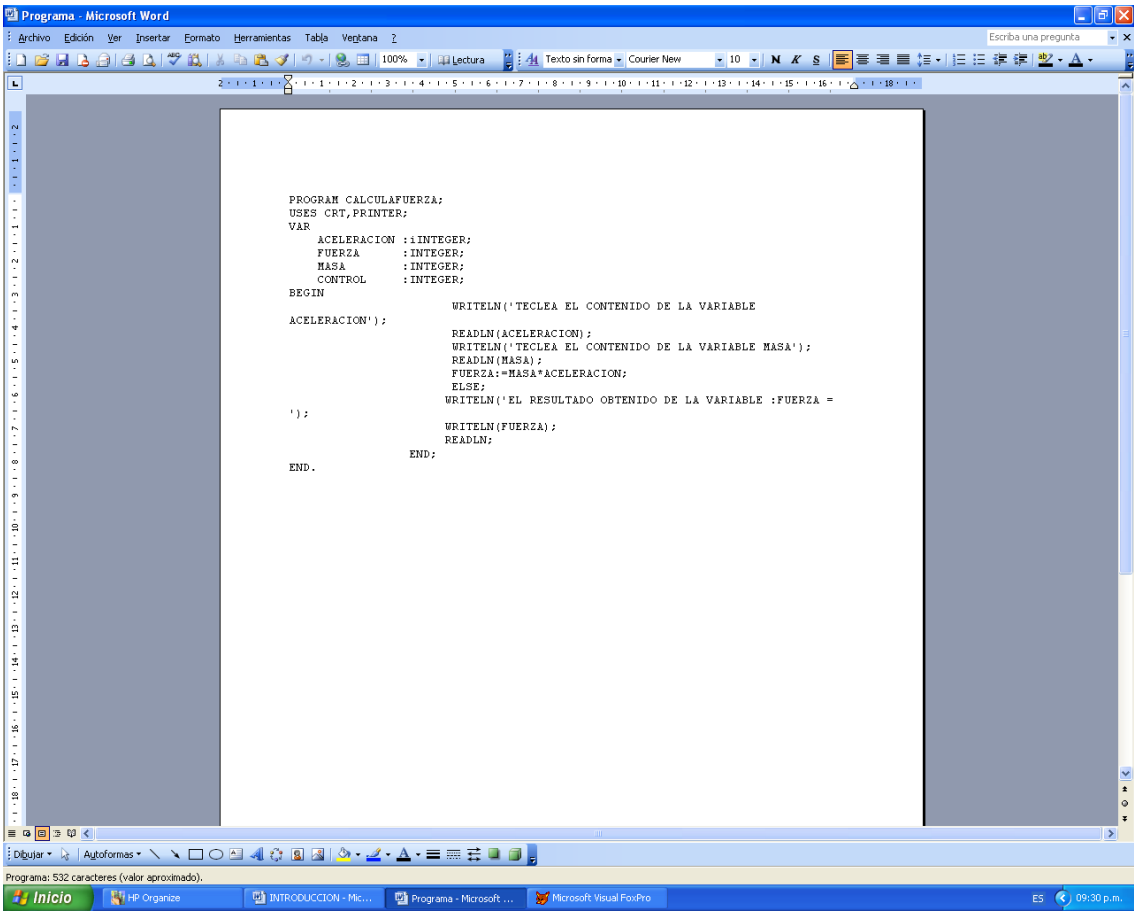


En el botón del disco, podrás guardar el archivo, para posteriormente utilizarlo y correrlo en pascal.

Da un clic en el botón salir.

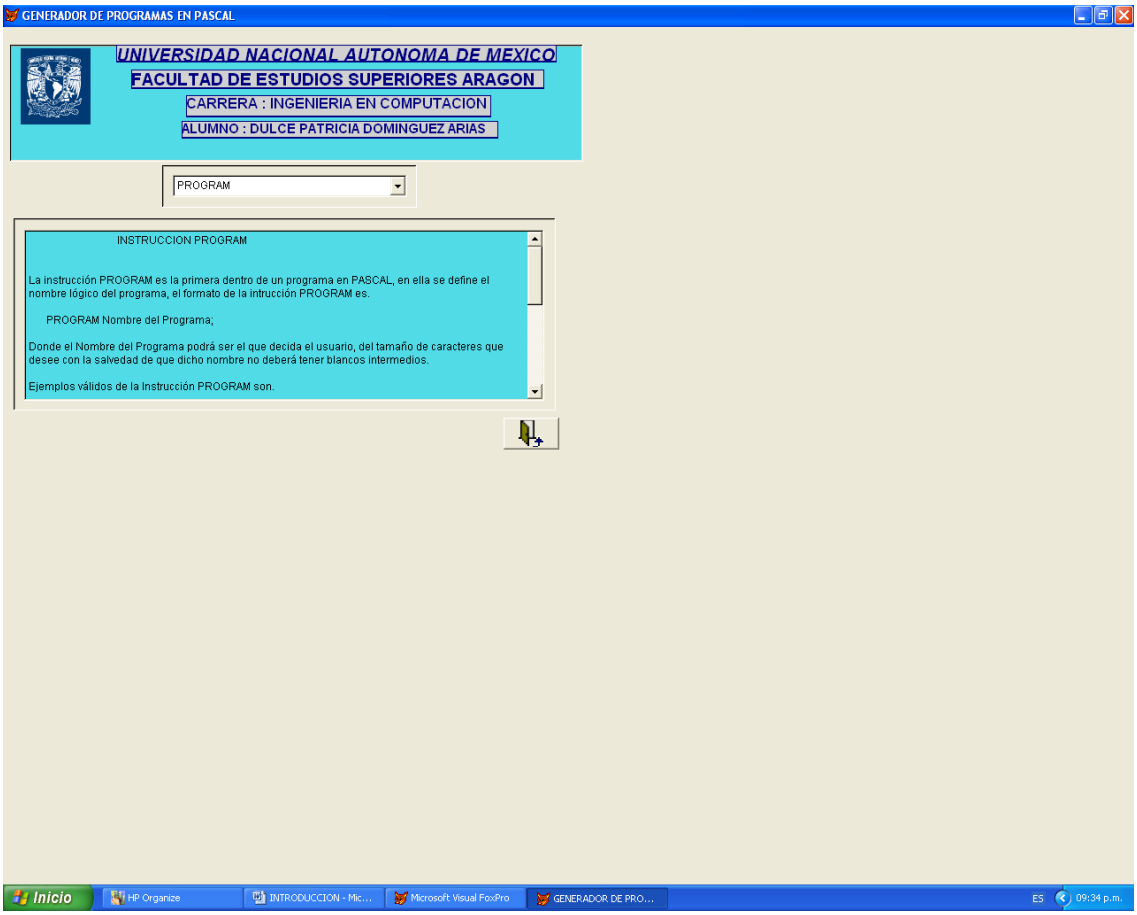


Para editar el programa en Word, haz clic en el botón de Microsoft Word.



Podrás ahora guardar el programa en la carpeta que quieras, y cambiarle el nombre si así lo deseas, el generador deja un archivo llamado programa.doc y otro llamado programa.pas, el cual podrás visualizar en el editor de pascal, modificarlo, compilarlo y ejecutarlo.

El botón de la linterna contiene un menú de ayuda por si necesitas realizar la consulta de alguna instrucción.



Cuando terminas, solo haz clic en el botón de salir, para cerrar la ayuda.

Haz clic de nuevo en el botón salir para cerrar el generador.

Podrás ahora ejecutar turbo pascal, visualizar tu archivo programa.pas y ejecutarlo.

Si se quisiera elaborar un programa que requiere algún ciclo dentro del mismo, deberá utilizarse la palabra REPITE con su respectivo índice de control (número de veces en las que se repetirá el ciclo) al inicio de la captura del algoritmo.

SECCIÓN DE MANTENIMIENTO DEL SISTEMA

Existen dentro del sistema llamado “Generador de Programas en Pascal”, una serie de bases de datos a las cuales es factible darles un mantenimiento, dependiendo de la actualización que se le quiera dar al mismo.

Dichas bases de datos son las siguientes:

- Base de datos Signo.- Engloba todo el conjunto de caracteres y símbolos especiales como , ; : , () + - * / ‘ ‘ “ “ = < > ^ etc. Esta base de datos, puede actualizarse conforme a las necesidades del usuario final y conforma el profesor que en este caso enseña el funcionamiento y aplicación del sistema

- Base de datos Reserva.- En esta base de datos se encuentra guardada la transformación de las palabras creadas dentro del pseudocódigo y convertidas a código de lenguaje de programación Pascal. Deberá realizarse la actualización, según se avance en la enseñanza de las instrucciones de Pascal.

- Base de datos Pascal.- En esta base de datos se encuentran todos los campo de tipo memo donde se describe el concepto y la sintaxis de todas las instrucciones del lenguaje de programación Pascal. Deberá realizarse también la actualización, según se avance en la enseñanza de las instrucciones de Pascal.

B L I B L I O G R A F I A

- Adoración de Miguel, Ed. AlfaOmega.
- Henry C. Lucas Jr. Sistemas de Información, Análisis, Diseño y Puesta a Punto. Ed. Paraninfo, S.A.
- Kendall & Kendall; Análisis y Diseño de Sistemas; 3a. Edición; Pearson Educación.
- López Leobardo, Turbo Pascal Ver. 7.0, Ed. Alfaomega, 1997, p.p 625.
- Martin James. Organización de Bases de Datos. Ed. Prentice Hall.
- Meter Abrams. Elementos de Procesos de Datos. Ed. C.E.C.S.A.
- Thompson, Phillip C. Círculos de Calidad. Cómo hacer que funcionen. Grupo Editorial Norma. Primera Edición. Colombia 1994.
- Varios. Guía de examen extraordinario de Cibernética y Computación II CCH Azcapotzalco.
- Varios. Programa de estudios de Cibernética y Computación I y II. UNAM. 2006.
- Páginas de Internet:
  - www.computacion
  - www.dgsca.unam.mx
  - www.uch.edu.ar/rrhh
  - www.visualfox
- Roger S. Presuman; Ingeniería del Software; 4a. Edición; McGraw Hill.