UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
ESCUELA NACIONAL
COLEGIO DE CIENCIAS Y HUMANIDADES
PLANTEL SUR
ACADEMIA DE MATEMÁTICAS

GUÍA PARA PREPARAR EL EXAMEN EXTRAORDINARIO DE LA ASIGNATURA DE CIBERNÉTICA Y COMPUTACIÓN II

SEMINARIO INSTITUCIONAL PARA DAR SEGUIMIENTO A LA APLICACIÓN DIDÁCTICA DE LOS PROGRAMAS DE TALLER DE CÓMPUTO Y DE CIBERNÉTICA Y COMPUTACIÓN

OCTUBRE 2005

INTRODUCCIÓN

La presente guía tiene como propósito orientarte en tu estudio para presentar con éxito el examen extraordinario de CIBERNÉTICA Y COMPUTACIÓN II.

En la guía encontrarás información acerca de los aprendizajes, estrategias de aprendizaje, contenido temario, actividades de autoevaluación y bibliografía que, conforme al programa de estudio vigente, serán considerados en el examen extraordinario de la asignatura.

Para que obtengas mejores resultados durante tu estudio es conveniente que utilices la guía de la siguiente manera:

TEMARIO DE ESTUDIO

En este listado identifica los temas que consideras que ya dominas, así como aquellos que desconoces o te resultan particularmente difíciles. Elabora un plan tentativo de trabajo, con días y cantidad de horas que dedicarás al estudio y repaso de los temas, para que logres los aprendizajes especificados en la guía.

ACTIVIDADES DE APRENDIZAJE

Realiza cada una de las actividades de aprendizaje sugeridas y elabora siempre un resumen o cuadro sinóptico destacando los aspectos más relevantes del aprendizaje. Esto te permitirá organizar tus conocimientos y ubicar los puntos principales de estudio y facilitará tus repasos. Ajusta tu plan inicial de trabajo de acuerdo a la importancia relativa a los aprendizajes, según tus aptitudes o dificultades para su adquisición.

De manera particular deberás prestar atención a lo siguiente:

Para el logro de los aprendizajes es recomendable que realices todas las actividades contenidas en la guía, esto te permitirá avanzar en forma gradual con los aprendizajes planteados en el programa de la asignatura. En cuanto a la resolución de problemas, aplica la metodología de solución de problemas y construye los programas relacionados con los problemas, con la finalidad de que obtengas aprendizajes sólidos sobre los fundamentos de la programación. Para ello, en la guía se anexa un CD ROM que contiene los programas que se mencionan en la guía y el compilador del lenguaje Freepascal para las plataformas Windows y Linux. Instala en tu computadora la versión para Windows. Para trabajar con el compilador, realiza lo siguiente c:\pp\bin\win32\fp. Asimismo, instala el compilador de Delphi desde Internet.

La guía contiene respuestas de algunas actividades de aprendizaje, las restantes contéstalas con base en tus conocimientos o bien con la consulta de la bibliografía recomendada. Además, contiene reactivos parecidos a los del examen, contéstalos y verifica tus aciertos con el listado de respuestas que se proporciona en la guía, para que detectes los aspectos que necesitas reforzar.

Recuerda que los reactivos de la guía son sólo indicativos del tipo de reactivos que puede contener el examen, y los aprendizajes explorados no excluyen otros (considerados en el programa de estudio vigente) no abordados en esta muestra.

BIBLIOGRAFÍA

Consulta para cada unidad los libros sugeridos en la guía. Puedes utilizar cualquier otro libro con el cual te sientas a gusto, ya que la bibliografía recomendada puede ser complementada y ampliada con libros que tú ya tengas o hayas utilizado anteriormente al estudiar estos temas.

TEMARIO DE ESTUDIO

El programa de la asignatura de Cibernética y Computación II, corresponde al Área de Matemáticas del Plan de Estudios vigente y contiene cinco unidades temáticas, las cuales se especifican a continuación:

Unidad I: Lenguaje de programación Pascal

Elementos del lenguaje.

- Estructura del programa.
  - ✓ Encabezado.
  - ✓ Zona de declaraciones.
  - ✓ Zona de sentencias.
- Identificadores, constantes, variables, palabras reservadas.
- Tipos de datos primitivos.
- Sentencias
  - ✓ Lectura, asignación, escritura.
  - ✓ Estructura de la sentencia condicional (If-then-else).
- Operadores, expresiones y orden de evaluación.
- Funciones y procedimientos estándar.

Unidad II. Estructura de control de secuencia

Estructuras de control de secuencia.

- Incondicionales.
  - ✓ Simples.
  - ✓ Compuestas.
  - ✓ La sentencia nula y el uso del terminador de sentencia (";")
- Sentencias condicionales:
  - ✓ Teorema de la programación estructurada.
- Estructura de Control.
  - ✓ *IF-THEN-ELSE*.
  - ✓ *WHILE-DO*.

Unidad IV. Estructuras de datos definidos por el usuario

Estructura de datos.

- Arreglos.
  - ✓ Elementos de un arreglo.
  - ✓ Índices y selección.
  - ✓ Almacenamiento de arreglos.
  - ✓ Dimensión de los arreglos.
  - ✓ Implementación.
- Cadenas de caracteres.
  - ✓ Implementación en el lenguaje de alto nivel.
  - ✓ Funciones y procedimientos de manejo de cadenas.
  - ✓ Conversiones a otros tipos de datos.
- Tipos enumerado y subrango.
- Conjuntos.
- Registros.
  - ✓ Elementos de un registro.
  - ✓ Campos y selección de elementos.
  - ✓ Almacenamiento de registros.
  - ✓ La sentencia With.
  - ✓ Registros variantes.

Manejo de archivos.

- Medios de almacenamiento secundario.
- Procesamiento secuencial.
  - ✓ Archivos de tipo texto.
  - ✓ Funciones y procedimientos para el manejo de archivos de tipo texto.

- ✓ *CASE*.
- ✓ *FOR-DO*.
- ✓ *REPEAT-UNTIL*.

.

Unidad III. Procedimientos y funciones

La modularidad como una herramienta para el manejo de programas.

- Concepto.
- Importancia.

Procedimientos y funciones.

- Concepto de procedimiento.
- Concepto de función.
  - ✓ Valor de regreso.
  - ✓ Tipos permitidos.
- Definición, declaración e invocación de procedimientos y funciones.
- Parámetros por:
  - ✓ Valor.
  - ✓ Referencia.
  - ✓ Variable.
- Alcance de identificadores.
  - ✓ Variables globales y locales.
  - ✓ Nombre de procedimientos y funciones.
- Recursividad.
  - ✓ Concepto Matemático
  - ✓ Implementación en el lenguaje.
  - ✓ Parte recursiva y parte terminal de un procedimiento recursivo**.**

- Procesamiento de tipo directo
  - ✓ Archivos de acceso directo.
  - ✓ Funciones y procedimientos para el manejo de archivos de acceso directo.

Estructuras dinámicas.

- Apuntadores.

Unidad V. Introducción a la programación en Delphi (Kylix)

Elementos de programación Delphi (Kylix).

- Objetos, eventos, diseño de ventanas, proyectos.
- Terminología básica, clases, instancias, herencia, polimorfismo.

Ambiente de trabajo.

- Pantalla principal.

Elementos básicos.

- Tablas.
- Formularios.
- Controles.
  - ✓ Cuadros de texto.
  - ✓ Etiquetas.
  - ✓ Botones.
  - ✓ Cajas de lista.
  - ✓ Cuadros combinados.
  - ✓ Botones de radio.
  - ✓ Casillas de verificación.
  - ✓ Contenedores.
  - ✓ Cuadros de imagen.
- Propiedades
  - ✓ Apariencia.
  - ✓ Comportamiento.
  - ✓ Posición.
  - ✓ Propiedades del control.
- Métodos
  - ✓ Mostrar y ocultar.
  - ✓ Actualizar.
  - ✓ Maximizar y minimizar.
  - ✓ Manipular el enfoque.
- Eventos.

- ✓ Inicialización y cierre de control.
- ✓ Recepción y pérdida del enfoque.
- ✓ Interacción con ratón.
- ✓ Interacción con teclado.

- Reportes.

UNIDAD I
LENGUAJE DE PROGRAMACIÓN PASCAL

INTRODUCCION. Al estudiar esta unidad tendrás la información suficiente para poder comunicarte con la computadora utilizando el lenguaje de programación Pascal, situación que te permitirá tener una visión más amplia de lo que es la programación no solamente en este lenguaje sino en cualquier otro.

PROPOSITOS. Al finalizar esta unidad podrás resolver problemas, usando los elementos del lenguaje de programación Pascal, mediante el desarrollo de programas.

Al finalizar esta unidad habrás alcanzado los siguientes aprendizajes:
Describirás la estructura del lenguaje de programación Pascal.
Conocerás la sintaxis y semántica de de las sentencias del lenguaje de programación Pascal.
Resolverás problemas que involucren funciones y procedimientos estándar mediante la ejecución de programas.

Para abordar los contenidos y alcanzar propósitos de esta unidad es necesario que recuerdes y apliques los conceptos aprendidos en las unidades 3 y 4 del programa de Cibernética y Computación I incluidos en la Metodología de la Resolución de Problemas entre otros el análisis del problema, el diseño de algoritmos que implica la construcción de diagramas de flujo y pseudocódigos, codificación, compilación, verificación, depuración y documentación de un programa.

El proceso de resolución de un problema con una computadora conduce a la escritura de un programa y a su ejecución en la misma. Aunque el proceso de diseñar programas es esencialmente un proceso creativo, se pueden considerar las etapas anteriores como las que primordialmente deben seguir todos los que se inician en el campo de la programación, independientemente del lenguaje utilizado.

Como recordarás la codificación consiste en escribir los pasos que se encuentran en el diagrama de flujo a instrucciones de algún lenguaje de programación, por ejemplo, Fortran, Algol, Cobol, Basic, Turbo c y Pascal entre otros.

Cabe señalar, que las instrucciones u órdenes para compilar y ejecutar programas en Pascal, puede variar según el tipo del compilador, en algunos casos compila y ejecuta con una sola orden, debido a que es un compilador de ambiente integrado y fuertemente amigable en el sentido de que es manipulado a través de menús y submenús; en otros casos las versiones del compilador necesitan instruir al sistema operativo, para que realice la face de ¨ montaje ¨ o enlace (link), carga del programa objeto con las librerías del programa del compilador. El proceso de ¨ montaje ¨, produce un programa ejecutable. Cuando el programa ejecutable se ha creado, se puede ejecutar desde el sistema operativo con solo teclear el nombre del programa.

Estructura de un programa en Pascal.

La codificación de cualquier programa, tiene que ver con las sentencias del lenguaje y la estructura de programas, en la mayoría de los lenguajes de alto nivel, en particular Pascal, tiene tres zonas, la primera se le conoce como la **zona de identificación de programas**, la segunda como la **zona para la declaración de datos** (unidades, directivas, variables, constantes, tipos de datos, funciones y procedimientos), que utilizan los programas y la tercera como la **zona para la declaración de sentencias** (instrucciones que corresponden a la parte ejecutable del programa).

Unidades en Pascal

El lenguaje de programación Pascal cuenta con varias unidades que permiten al programador un completo dominio del control de los periféricos de entrada y salida de la computadora, así como las rutinas de graficación, entre otros.

Estrategia de aprendizaje: Investiga y contesta las siguientes preguntas:

¿Qué es una unidad en Pascal? _____
_____

¿Cuáles son las unidades de Pascal? _____
_____

¿Qué es la unidad (unit) CRT? _____
_____

¿Cuáles son las funciones y procedimientos de la unidad CRT? _____
_____

Cuando se carezca de la información suficiente para la construcción de programas en Pascal se puede solicitar ayuda en línea para lo cual es necesario conocer la estructura del ambiente integrado de Pascal ya que de cada uno de ellos tiene formas diferentes de hacerlo, sin embargo es importante que sepas que siempre existe ayuda en línea.

Identificadores y palabras reservadas en Pascal

Los **identificadores** en Pascal son los nombre que damos a constantes, tipos, variables, funciones, procedimientos, unidades, programas y campos que forman un registro, tienen reglas especificas, entre otras, empezar con una letra, no contener espacios ni caracteres de puntuación (‘, ;, ?, etc.) y tienen una longitud de hasta 64 caracteres, las **palabras reservadas** que son aquellas que tienen un significado específico para Pascal, no pueden ser utilizada como identificadores.

Estrategia de aprendizaje: ¿Cuál de los siguientes nombres sirven como identificadores?

_____

_____

robocop	125años	módulo	inventario	salario_actual
*mijuego	migrafica	impuesto	velocidad	procedure
repeat	dista*ncia	bin_hexa	begin	amx127

Estrategia de aprendizaje: Investiga y escribe las palabras reservadas en el lenguaje de programación Pascal:

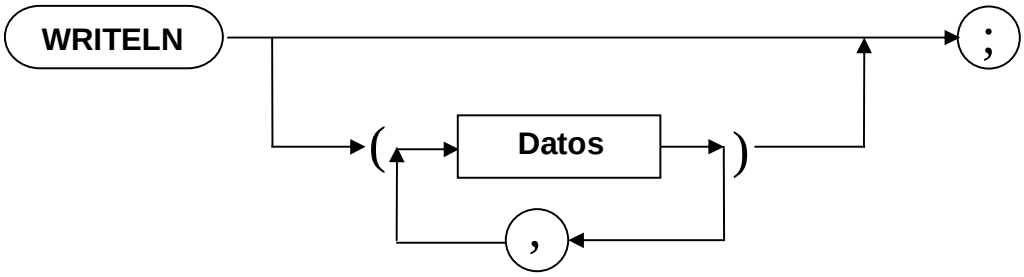
_____

Instrucciones en Pascal

Recuerda que el algoritmo, pseudocódigo o diagrama de flujo, indican la secuencia de pasos lógicos necesarios para la resolución del problema sin importar el lenguaje en el que se construya el programa, particularmente en Pascal existen instrucciones que describen procesos identificados plenamente por el compilador, a estas instrucciones se les denomina sentencias, dentro de las cuales están:

Sentencia de escritura **WRITELN**

Esta sentencia se utiliza para escribir en algún dispositivo de salida números, letras, títulos y símbolos del código ASCII. A continuación se presenta el diagrama sintáctico.



Este diagrama sintáctico de la sentencia WRITELN, indica la sintaxis y semántica que debe seguirse para la escritura de datos, para ello se recorre el diagrama como lo indican las flechas, en este recorrido, lo que está en el interior del símbolos óvalo y círculo se escriben igual y lo que está en el rectángulo representa a los datos de salida que serán visualizados, terminando con el carácter punto y coma (;).

Ejemplos: sentencia de escritura	Visualización
WRITELN;	Línea en blanco
WRITELN(‘HOLA AMIGA’);	HOLA AMIGA
WRITELN(‘ ‘,’DEJA 5 ESPACIOS’);	DEJA 5 ESPACIOS
SUMA:=5000;	
WRITELN(‘SUMA = ‘,SUMA);	SUMA = 5000

La estructura de programas en Pascal, se da en la codificación de un programa que te permitirá construir en pantalla una carátula con algunos datos, utilizando la sentencia WRITELN.

```
program caratula     {Comentario:  zona para la identificación del programa}
begin                {Comentario: inicia zona para la declaración de sentencias}
   writeln('UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO');
   writeln('COLEGIO DE CIENCIAS Y HUMANIDADES');
   writeln('PLANTEL SUR');
   writeln('GUIA DE CIBERNÉTICA Y COMPUTACIÓN II');
   writeln('MANEJO DEL EDITOR DE PASCAL');
   readln;
end.                 {Comentario: termina zona para la declaración de sentencias}
```

En la codificación anterior, **program**, **begin** y **end** son **palabras reservadas**, utilizadas en el lenguaje de programación Pascal para identificar, delimitar el inicio y fin de un programa, respectivamente, caratula es el nombre del programa y la sentencia **readln**, se utiliza para hacer una pausa en la ejecución del programa y permite visualizar la salida del mismo, para continuar con la ejecución, se oprime la tecla "enter". Como podrás observar, este programa no tiene la zona de declaración de datos ya que no los contiene.

Edición de un programa en Pascal.

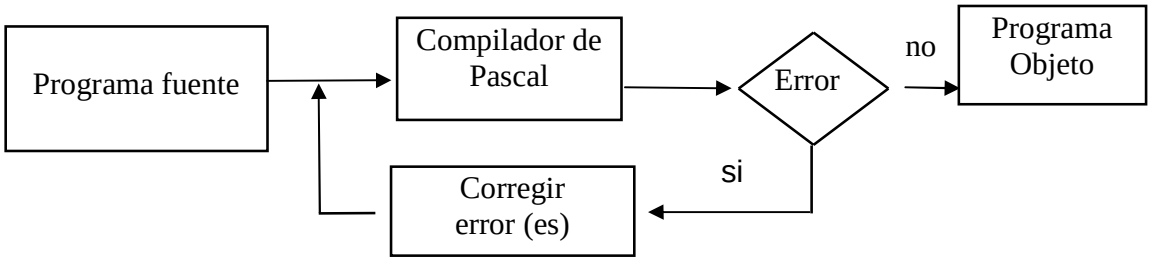
La edición de un programa, consiste en introducirlo con el teclado a la memoria RAM de la computadora por medio de un editor o procesador de textos, en particular, utilizaremos el editor del compilador de Pascal. De manera parecida se hace la carga del ambiente integrado del compilador de Pascal.

Estrategia de aprendizaje: Edita el programa y sálvalo con el nombre **carat1.pas**.

Compilación de un programa en Pascal

Una vez que el programa se encuentra en memoria de la computadora, está listo para ser compilado, ¿escribe en qué consiste este importante proceso? _____
_____

Utilizando la frase "un diagrama dice más que mil palabras", al proceso de compilación, se ilustra por medio del siguiente diagrama:



Cualquier compilador, en particular el de Pascal, acepta como entrada un programa fuente (como carat1.pas), lo traduce al lenguaje de máquina, para checar la sintaxis de las sentencias y declaraciones del programa. Si ningún error es detectado, genera como salida el programa objeto escrito en lenguaje de máquina, listo para su ejecución, en caso contrario, el compilador lo despliega en la pantalla de trabajo y no se genera el programa objeto, para obtenerlo, corrige los errores detectados y compila el programa hasta que quede exento de errores.

Estrategia de aprendizaje. Compila el programa carat1.pas.

Ejecución de un programa en Pascal

Una vez que se obtiene el programa objeto como resultado del proceso de compilación, se procede a su corrida.

Estrategia de aprendizaje: Ejecuta el programa carat1.pas y contesta la pregunta.

¿Qué visualizas en la pantalla? _____
_____

La salida del programa caratula, se obtuvo con la sentencia WRITELN en el formato por default de ésta, es decir, la escritura de los títulos empezando en el primer renglón, a renglón seguido y empezando al principio de cada línea de la pantalla, sin embargo, estarás de acuerdo que hace falta centrar los títulos de la carátula, recuerda que el compilador no es un procesador de texto, por lo que se tiene que modificar el programa caratula, para que los títulos queden centrados en pantalla, por ahora, guarda tu programa.

Centrado de títulos en la pantalla.

Esta actividad, tiene como finalidad, ampliar tus conocimientos en el manejo del ambiente integrado de Pascal y de la sentencia de escritura **WRITELN** con un formato de salida diferente al de default.

Estrategia de aprendizaje: Contesta las siguientes preguntas

¿Qué es la pantalla del monitor? _____
_____

¿Cuántas columnas tiene la pantalla en modo texto? _____

¿Cuántos renglones tiene la pantalla en modo texto? _____

¿Capacidad de la pantalla en modo texto para visualizar información? _____

¿Cuál es el proceso para centrar el primer título del programa carta1.pas? _____
_____

Existen diferentes formas para centrar un título en Pascal, como son: dejar espacios en blanco, declarar espacios en blanco y utilizar la sentencia GOTOXY(x,y).

Ejemplo: Para centrar el título UNAM, detallaremos cada una de las formas:

Sentencias de escritura	Visualización en pantalla
WRITELN(‘ ‘,’UNAM’);	UNAM
WRITELN(‘ ‘:20,’UNAM’);	UNAM
GOTOXY(20,1);	
WRITELN(‘UNAM’);	UNAM

En la primera sentencia de la tabla, se dejan veinte espacios delimitados por el apóstrofe (‘), por lo que se visualiza en pantalla el título UNAM en la columna 21, en la segunda sentencia hay un espacio delimitado por el apóstrofe (‘) y el número 20 indica los espacios que se dejan antes de escribir el título UNAM y se visualiza lo mismo que en la primera, la tercera indica la posición de la columna y renglón en la pantalla y en la cuarta se visualiza el mismo título de las dos primeras.

La sentencia GOTOXY(x,y) es un procedimiento contenido en la unidad CRT y especifica la posición del cursor en la pantalla, por ejemplo GOTOXY(10,5) en la columna 10 y renglón 5. Para que sea reconocida por el compilador, inserta la cláusula USES CRT, inmediatamente abajo del nombre de programa.

Para centrar los restantes títulos, se procede en forma análoga al centrado del primer título.

Estrategia de aprendizaje: Escribe en la tabla siguiente, el número de espacios que se dejarás antes de escribir cada título del programa carat1.pas.

Título	1	2	3	4	5
No. de espacios					

Estrategia de aprendizaje: Edita el programa carat1.pas, incluyendo los espacios de la tabla anterior, para que los títulos queden centrados, tal como se indica a continuación:
a) Dejando los blancos necesarios y grábalo con el nombre de archivo carat2.pas.
b) Usando la sentencia GOTOXY(x,y) y grábalo con el nombre de archivo carat3.pas.

Estrategia de aprendizaje: Compila y ejecuta el programa carat2.pas y contesta la pregunta.

¿Qué visualizas en la pantalla? _____
_____

Estrategia de aprendizaje: Compila y ejecuta el programa carat3.pas y contesta la pregunta.

¿Qué visualizas en la pantalla? _____
_____

Estrategia de aprendizaje: Edita el programa carta1.pas con la sentencia GOTOXY(x,y), para que la carátula quede centrada a lo largo y ancho de la pantalla. Graba el programa con el nombre de archivo carat4.pas, compílalo, ejecútalo y contesta la pregunta:

¿Qué visualizas en la pantalla? _____
_____

Como podrás observar, la salida de los programas carat2, carat4 y carat4 se obtuvo centrado los títulos, sin embargo, estarás de acuerdo que hace falta hacer una mejor presentación, lo cual lograrás conseguir mediante algunos procedimientos.

Funciones y Procedimientos en Pascal

Estrategia de aprendizaje: Investiga y contesta las siguientes preguntas:

¿Qué es en Pascal, una función estándar?_____
_____

¿Qué es en Pascal, un procedimiento estándar?_____
_____

Estrategia de aprendizaje: Describe los procedimientos:

CLRSCR:
_____
_____

DELAY:
_____
_____

TEXTBACKGROUND:
_____
_____

TEXTCOLOR:
_____
_____

Estrategia de aprendizaje: Con la finalidad de ilustrar los procedimientos anteriores, salva el programa carat4.pas con el nombre de archivo carat5.pas, edita este archivo, como el programa que se presenta a continuación:

```
program carat5;          {Identificación del programa}
uses crt;                {Declaración de unidades}
begin
        TEXTBACKGROUND(WHITE);
        CLRSCR;
        TEXTCOLOR(RED);
        GOTOXY(20,6);
        WRITELN('UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO');
        GOTOXY(23,7);
        WRITELN('COLEGIO DE CIENCIAS Y HUMANIDADES');
         GOTOXY(34,8);
         WRITELN('PLANTEL SUR');
         GOTOXY(21,14);
         WRITELN('MATERIA: CIBERNÉTICA Y COMPUTACIÓN II');
         GOTOXY(21,15);
         WRITELN('MANEJO DEL EDITOR DE PASCAL');
         DELAY(20000);
end.
```

Colores que se utilizan en TEXTBACKGROUND y TEXTCOLOR:

Constante	Color
0	Negro
1	Azul
2	Verde
3	Azul claro
4	Rojo
5	Magenta
6	Marrón
7	Gris claro
8	Gris oscuro
9	Azul claro
10	Verde claro
11	Azul claro
12	Rojo claro
13	Magenta claro
14	Amarillo
15	Blanco
16 - 31	Colores parpadeantes
128 + color	Color parpadeo.

Estrategia de aprendizaje: Edita el programa carat5.pas con otros colores a TEXTBACKGROUND y TEXTCOLOR de la tabla anterior. Salva, compila y ejecuta el programa mencionado.

Actividades de aprendizaje. Considera las siguientes palabras y escribe en el paréntesis **id** para identificador, **pr** para palabra reservada y **ni** para ninguno de ellos.

nómina ( )	nomina ( )	2caratula ( )	carat2 ( )
begin ( )	amx125 ( )	distancia ( )	pago_neto ( )
media ( )	#gato ( )	until ( )	juego ( )
inventario ( )	procedure ( )	área ( )	perímetro ( )

Declaración de variables en el lenguaje de programación Pascal

La elaboración de los programa de carátula en sus diversa salidas, permitió un primer acercamiento al manejo del ambiente integrado de Pascal y de algunos elementos básicos de la programación como la estructura de programas en Pascal, identificadores, palabras reservadas, unidades, procedimientos de la unidad CRT y la sentencia de escritura en formato diferente al de default.

Los siguientes dos problemas tienen como finalidad reforzar los conceptos abordados y ampliar el conocimiento del manejo del ambiente integrado de Pascal, así como la declaración de constantes, variables y sus tipos, sentencias de lectura y asignación.

Estrategia de aprendizaje: Investiga y contesta las siguientes preguntas:

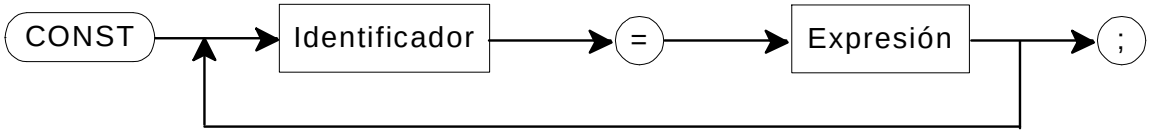
¿Qué es una constante? _____

¿Qué es una variable? _____

Declaración de datos en Pascal

El lenguaje de programación Pascal es fuertemente tipificado, lo que obliga al programador a informar al compilador de los tipos de datos que serán utilizados en el programa. Los lenguajes de alto nivel tienen reglas bien definidas para la declaración de datos y sus tipos, abordaremos las correspondientes de Pascal, mediante diagramas sintácticos.

Diagrama sintáctico para la declaración de constantes:



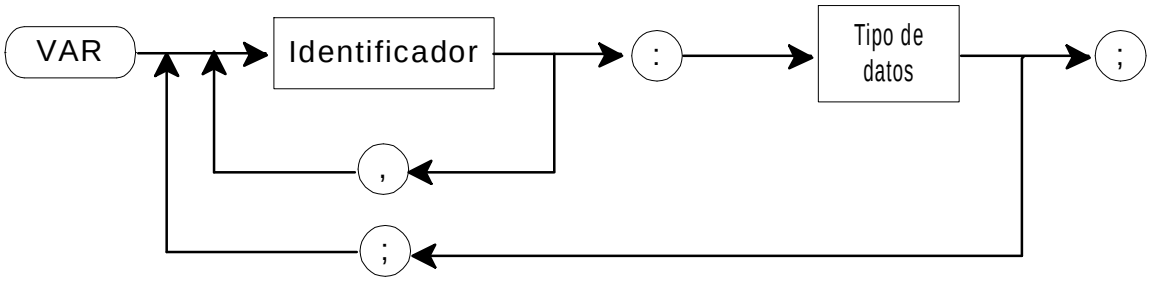
Este diagrama sintáctico, indica la sintaxis y semántica en que deben ser declaradas las constantes, para ello se recorre el diagrama como la indican las flechas, en este recorrido lo que está dentro del símbolo óvalo y círculo se escribe igual, lo que está en el primer rectángulo, representa el identificador de la constante y será declarado en la zona de

declaraciones del programa, lo que está en el segundo rectángulo es el valor de la expresión con la cual quedará definida la constante.

Así, para ilustrar la declaración de constantes, se presenta el siguiente ejemplo:

```
const
     costolitrogas =4.65;
    rendauto =7.5;
    titulo =‘Calculo del costo’;
    caracter =‘a’;
    limite =157;
```

Diagrama sintáctico para declarar variables.



Este diagrama sintáctico, indica la sintaxis y semántica en que deben ser declaradas las variables, para ello se recorre el diagrama como lo indican las flechas, en este recorrido lo que está dentro del símbolo óvalo y círculo se escribe igual, lo que está en el primer rectángulo, representa el identificador de variable que será declarado en la zona de declaración de datos y lo que está en el segundo rectángulo representa el tipo de la variable.

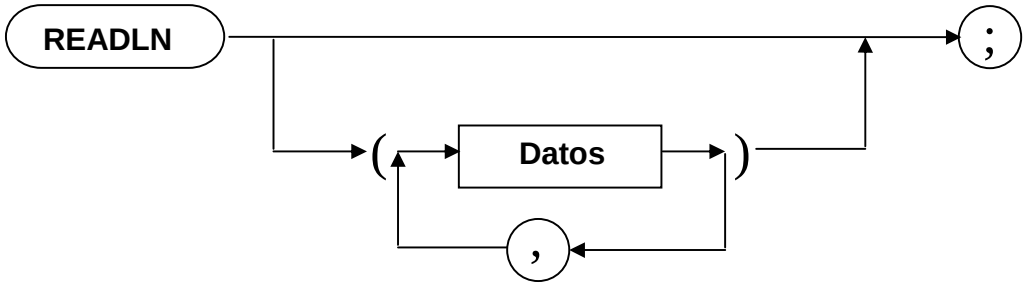
En esta unidad se tratarán únicamente datos de tipo primitivo: entero (**integer**), real (**real**), carácter (**char**) y booleano (**boolean**).

Así, para ilustrar la declaración de variables, se presenta el siguiente ejemplo:

```
var
    num1, num2, suma, resta: integer;
    consumogas, gasto_viaje:real;
    velocidad_ media: integer;
    distancia: real;
```

Sentencia de lectura READLN

Esta sentencia permite la lectura de datos desde cualquier periférico de entrada, el teclado u disco, para un mejor conocimiento de ella, se da su diagrama sintáctico.



Este diagrama sintáctico de la sentencia READLN, indica la sintaxis y semántica que debe seguirse para la lectura de datos, para ello se recorre el diagrama como lo indican las flechas, en este recorrido, lo que está en el interior del símbolos óvalo y círculo se escriben igual y lo que está en el rectángulo representa los datos de entrada al programa, termina la introducción de datos con el carácter de punto y coma (;).

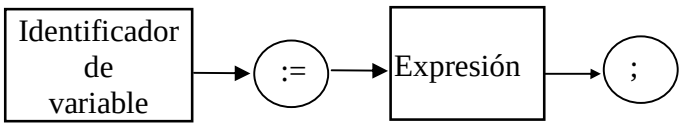
Ejemplo: Para lectura de datos, se tiene:

Ejemplos: sentencia de lectura	Función de la sentencia
READLN;	Pausa en la ejecución
READLN(distancia);	Pausa para introducir el valor de distancia
READLN(num1, num2);	Pausa para introducir los valores de num 1 y 2.
WRITE('Introduce costo: ');	Escritura de Introduce costo:
READLN(costo);	Pausa para introducir el valor de costo,
	inmediatamente, después del título anterior.

Sentencia de asignación :=

Esta sentencia permite asignar valores a las variables utilizadas en el programa, para una mejor comprensión de la sintaxis y semántica de la misma, se presenta su diagrama sintáctico.

Diagrama sintáctico de la sentencia de asignación :=



Codificación
variable := expresión;

Este diagrama sintáctico, indica la sintaxis y semántica que debe seguirse para la asignación de valores a las variables especificadas en el programa, para ello se recorre el diagrama como lo indican las flechas, en este recorrido, lo que está en el interior del símbolo círculo se escriben igual y lo que está en el primer rectángulo representa a la variable que se le asignará el valor de expresión contenida en el segundo rectángulo, la asignación termina con el carácter de punto y coma (;).

En la asignación de valores a variables, el tipo de variable y el de expresión deberán ser compatibles (que sean del mismo tipo), de lo contrario se originará un error por la incompatibilidad de tipos, por otro lado no deberá ser considerada como una igualdad.

Ejemplo para se ilustrar la sentencia de asignación:

```
var
      num1, num2, suma, entero: integer;
      cociente: real;
      caracter: char;
      logico: boolean;
begin
       WRITE('Digita num1: ');
       READLN(num1);
       WRITE('Digita num2: ');
       READLN(num2);
       suma:=num1+num2;
       cociente:=num1/num2;
       logico:=true;
       caracter:='t';
end.
```

Problema viaje. Encuentra el tiempo y el costo de un viaje en automóvil que la familia Sánchez, realiza del Distrito Federal al Puerto de Acapulco (tiempo, costo), durante el recorrido pagará en total $ 450 de cuotas en las casetas ubicadas en el trayecto. Además, se sabe que la distancia entre ambas ciudades es de aproximadamente 400 kilómetros (distancia), la velocidad promedio del auto es de 90 kilómetros por hora (velocidad). El rendimiento del automóvil (rend) y el costo por litro de gasolina (costo_litrogas) deberán ser introducidas con el teclado. Se entiende por rendimiento del automóvil a los kilómetros recorridos por litro de gasolina.

Estrategia de aprendizaje: Analiza e identifica cada uno de los elementos del problema:

Datos de entrada	*tipo*	*Datos de salida*	*tipo*
_____	_____	_____	_____
_____	_____	_____	_____

Constantes	*Procesos (operaciones)*
_____	_____
_____	_____

Actividad de aprendizaje. Escribe en las rayas, las sentencias faltantes en el pseudocódigo.

```
Inicio
  Escribir('Rendimiento: ')
  Leer(rend);
  Escribir('Costo litro gasolina: ')
  Leer(costo_litrogas);
  tiempo ← _____________________
  consumo_gas ← _________________
  costo ← consumo_gas*costo_litrogas
  Escribir('Tiempo viaje:  ',tiempo:5:2)
  Escribir('total gas: ',consumo_gas:5:2)
  Escribir('costo = ',costo_viaje:5:2)
Fin
```

Actividad de aprendizaje. Escribe en las rayas, las sentencias faltantes en la codificación.

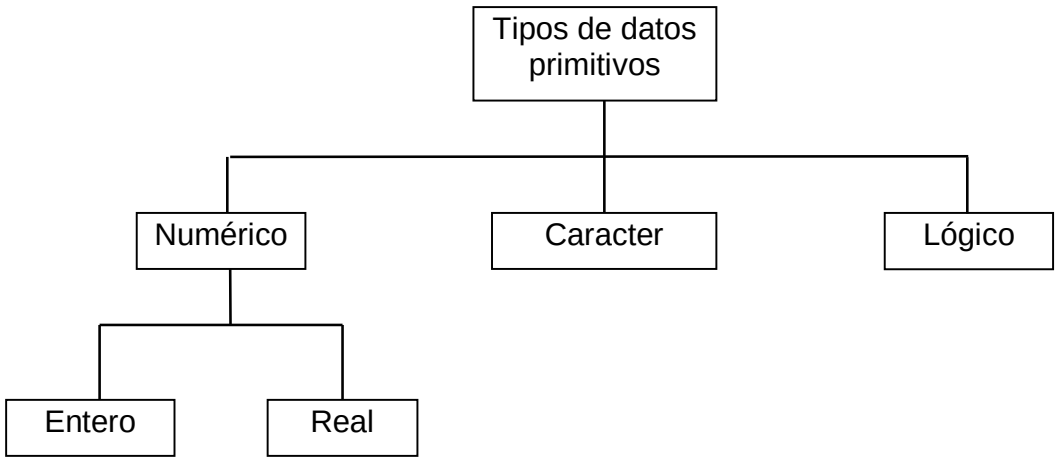
```
program viaje;
uses crt;
const
   dist = 400;
   vm   = 90;
var
  rend:integer;
 costo_litrogas, consumo_gas, costo:real;
begin
    clrscr;
    write('Rendimiento: ');
    _____________________________
    write('Costo litro gasolina: ');
    _____________________________
    tiempo:=distancia/velocidad;
    consumo_gas:=distancia/rend;
    costo:= _____________________
    writeln('total gas: ',consumo_gas:5:2);
    writeln('costo = ',costo:5:2);
    readln;
end.
```

Actividad de aprendizaje. Realiza la edición y graba el programa con el nombre de archivo viaje.pas. Compílalo, ejecútalo con los datos de la tabla y completa los datos de la misma.

rend	costo_litrogas	tiempo	consumo_gas	costo
12	6.50			
8	6.75			
10	6.80			

Tipos de datos primitivos

Como se mencionó, en esta unidad se tratarán únicamente datos de tipo primitivo, los cuales son: entero (**integer**), real (**real**), carácter (**char**) y booleano (**boolean**). El siguiente diagrama, representa a los tipos de datos primitivos:



Datos de tipo numérico:

Tipo	Rango	Formato
byte	0 .. 255	8 bits sin signo
integer	-32768 .. 32767	16 bits con signo
longint	-247483648 .. 2147483647	32 bits con signo
shortint	-128 .. 127	8 bits con signo
word	0 .. 65535	16 bits sin signo

Tipo	Rango	Cifras	Tamaño y bytes
real	$2.9x10^{-39}$ .. $1.7x10^{38}$	11 - 12	6
single	$1.5x10^{-45}$ .. $3.4x10^{38}$	7 - 8	4
double	$5.0x10^{-324}$ .. $1.7x10^{308}$	15 - 16	8
extended	$1.9x10^{-4931}$ .. $1.1x10^{4932}$	19 - 20	10
comp	-2^{63} .. $2^{63} - 1$	19 - 20	8

Datos de tipo caracter*: char*

-Cualquier caracter del código ASCII.

-Una cadena (**string**) es un sucesión de caracteres que se encuentran delimitados por apóstrofos.

-Datos de tipo lógico (**boolean**), es aquel que puede tomar uno sólo valor de dos posibles (verdadero u falso).

Operadores, expresiones y orden de evaluación.

Una expresión es una combinación de operandos y operadores, según el tipo de ellos, se clasifican en aritméticas, relacionales, lógicas y carácter.

Operadores aritméticos básicos

Operador	Significado	Ejemplo	Resultado
+	Suma	a + b	Suma de a y b
-	Resta	a - b	Diferencia de a y b
*	Multiplicación	a * b	Producto de a por b
/	División	a / b	Cociente de a por b
div	División entera	a div b	Cociente entero de a / b
mod	Módulo	a mod b	Resto entero de a / b

Reglas de evaluación de expresiones (prioridad)

1. Todas las subexpresiones entre paréntesis se evalúan primero. Las subexpresiones con paréntesis anidados se evalúan de dentro hacia afuera; el paréntesis más interno se evalúa primero.

2. Prioridad de operaciones. Dentro de una misma expresión o subexpresión, los operadores se evalúan en el siguiente orden:

*, /, div, mod. Se evalúan primero.
+, -. Se evalúan al final.

3. Regla asociativa izquierda. Los operadores en una misma expresión o subexpresión con igual nivel de prioridad, se evalúan de izquierda a derecha.
Ejemplos:
1. 23 + 3*5 Primero se realiza la multiplicación, después la suma.
23 + 15 = 38

2. 13 * 3 div 7 Primero se realiza el producto, después div.
39 div 7 = 5
3. 45 mod 6 - 10 Primero se realiza mod, después la resta.
3 - 10 = -7

4. 5 + 6*(15 - (5 + 7)) Primero el paréntesis más interno.
5 + 6*(15 - 12) Después el menos interno.
5 + 6*(3) Después el producto
5 + 18 = 23 Por último la suma.

5. (7 - 4)*8 div 3 mod 5 - 9 Primero se elimina el paréntesis
3 * 8 div 3 mod 5 - 9 Después el producto
24 div 3 mod 5 - 9 Después div
8 mod 5 - 9 Después mod
3 - 9 Después suma

Escritura de fórmulas en Pascal

Fórmulas matemáticas	Expresión en Pascal
area = $\frac{ba}{2}$	area:= (b*a)/2;
f = $\left(\frac{9}{5}\right)c + 32$	f:= (9/5)*c + 32;
pendiente = $\frac{y_2 - y_1}{x_2 - x_1}$	pendiente:= (y_2 - y_1)/(x_2 - x_1);
distancia = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	distancia := sqrt (sqr(x_2-x_1)+(y_2-y_1));
x = $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$	x:=(-b+sqrt(b*b-4*a*c))/(2*a);
x = r cos x	x:=r*cos(x);
y = r sin x	y:=r*sin(x);

Funciones estándar
Las funciones son uno de los elementos básicos en la programación y son de dos tipos las estándar y las definidas por el usuario, esta sección se trabaja con las primeras, en la unidad tres abordaremos a las segundas.

Una función estándar no necesita ser declarada y codificada en el programa, para ser utilizada en las expresiones aritméticas, solo tenemos que llamarla con su nombre y argumentos, tal como se ilustran en los ejemplos de arriba.

Estrategia de aprendizaje: Investiga y describe las siguientes funciones estándar:

Nombre	Argumento (x)	Descripción
abs(x)		
arctan(x)		
cos(x)		
exp(x)		
ln(x)		

Pi		
round(X)		
Sin(x)		
sqr(x)		
sqrt(x)		
trun(x)		
ord(x)		
chr(x)		
succ(x)		
pred(x)		

Problema funciones. Calcular la distancia, pendiente y el ángulo de inclinación de la recta que forman los puntos P1(x1, y1) y P2(x2, y2) en el plano cartesiano XY. Asimismo, determinar el valor de las funciones seno, coseno y tangente del ángulo de inclinación.

Estrategia de aprendizaje: Analiza e identifica los elementos del problema

Datos de entrada: *tipo*

_____ _____
_____ _____
_____ _____
_____ _____

Datos de salida: *tipo*

_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____

Procesos:

dist ← _____
pend ← _____
tang ← _____
seno ← _____
coseno ← _____
tang ← _____

Estrategia de aprendizaje. Completa las líneas faltantes en el pseudocódigo.	Estrategia de aprendizaje. Completa las líneas faltantes en la codificación.
Inicio escribir('Digita las coordenadas de P1: ') leer(x1,y1); escribir('digita las coordenadas de P2: ') _____ dist ← $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ pend ← _____ ang ← arcotangente(pend) seno ← sin(ang); coseno ← cos(ang); x ← x2-x1 y ← y2-y1 tang ← _____ escribir('Distancia = ',dist:5:2) escribir('Pendiente = ',pend:5:2) escribir('Ángulo = ', ang*180/pi:5:2) escribir('Seno = ',seno:5:2); escribir('Coseno = ',coseno:5:2) _____ Fin	program funciones; uses crt; var x1,y1,x2,y2:integer; dist,pend,ang:real; seno,coseno,tang:real; _____ begin clrscr; write('Digita las coordenadas de P1: '); readln(x1,y1); _____ _____ dist:= _____ pend:= _____ ang:=arctan(pend); seno:=sin(ang); coseno:=cos(ang); x:=x2-x1; y:=y2-y1; tang:= _____ writeln('Distancia = ',dist:5:2); writeln('Pendiente = ',pend:5:2); writeln('Ángulo = ', ang*180/pi:5:2); writeln('Seno = ',seno:5:2); writeln('Coseno = ',coseno:5:2); write('Tangente = ', tang:5:2); readln; end.

Estrategia de aprendizaje:

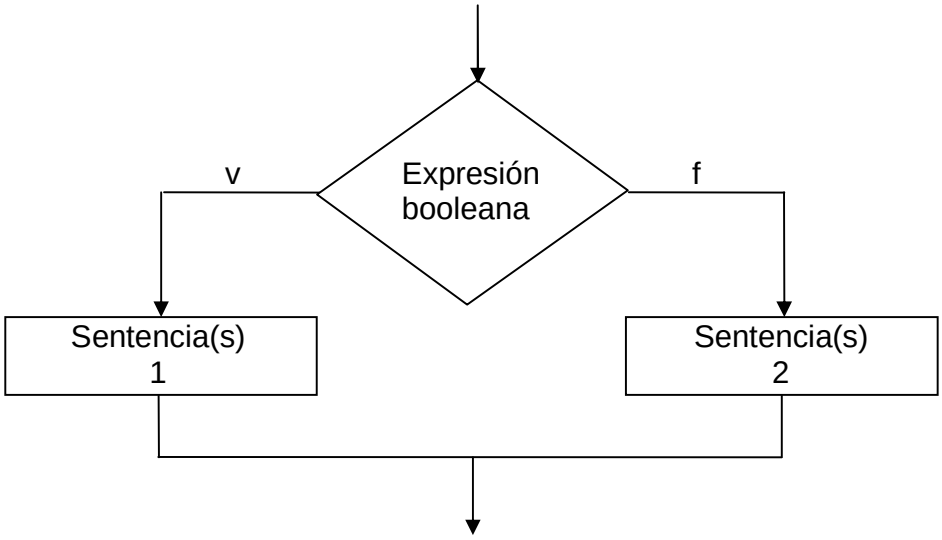
a) Edita el programa funciones y grábalo con el nombre de archivo funcion1.pas. Asimismo,
compila y ejecuta el programa varias veces con diferentes datos de entrada.

b) Modifica el programa funcion1.pas con la sentencia tang:=seno/coseno; en lugar de la sentencia tang:=y/x; grábalo con el nombre de archivo funcion2.pas, compílalo y ejecútalo
varias veces con diferentes datos de entrada.

Estructura de la sentencia IF-THEN-ELSE

Esta sentencia cuya utilidad es la de poder seleccionar una alternativa de entre dos posibles.

Diagrama sintáctico de la sentencia IF-THEN-ELSE



Este diagrama de flujo de la sentencia IF-THEN-ELSE especifica el algoritmo que deberá seguirse para seleccionar una alternativa entre dos posibles, para ello, se recorre el diagrama como lo indican las flechas en este recorrido, primero se evalúa la expresión booleana, si la evaluación es verdadera, se ejecutan las acciones contenidas en Sentencia(s) 1, de otro modo, se ejecutan las contenidas en Sentencia(s) 2

Problema (día de la semana). Se desea asignarle un número a cada día de la semana, desde 1(domingo) hasta 7(sábado).

Estrategia de aprendizaje: Analiza e identifica los elementos del problema

Datos de entrada: *tipo* *Datos de salida:* *tipo*

_____ _____ _____ _____

Procesos:

_____

Estrategia de aprendizaje: Completa el pseudocódigo del problema día de la semana:

```
Inicio
    Leer (dia)
    Si (dia = 1) entonces
        Escribir ('1: Domingo')
    Sino
        Si (dia = 2) entonces
            Escribir ('2: Lunes')
        Sino
            Si (dia = 3) entonces
                Escribir ('3: Martes')
            Sino
                Si (dia = 4) entonces
                    Escribir ('4: Miércoles')
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                _____________________________
Fin
```

Estrategia de aprendizaje: Completa la codificación del problema día de la semana:

```
program dia_semana;
uses crt;
  var
      dia:integer;
begin
      clrscr;
      write('Digita el día de la semana');
      readln(dia);
      if (dia=1) then
          writeln('1:Domingo')
      else
         if (dia=2) then
            writeln('2: Lunes')
      else
         if (dia=3) then
            writeln('3: Martes')
```

```
    else
        if (dia=4) then
            writeln('4: Miércoles')
        else
            _____________________________
            _____________________________
            _____________________________
            _____________________________
            _____________________________
            _____________________________
            _____________________________
            _____________________________
                else
                    writeln('Día fuera de rango');
    readln:
end.
```

Estrategia de aprendizaje: Edita el programa dia_semana, grábalo con el nombre de archivo dias.pas, compílalo y ejecútalo para varias veces.

Reactivos muestra

1. La zona de la estructura de un programa en Pascal donde se hace la declaración de unidades, variables, constantes, tipos de datos, funciones y procedimientos, es:
son:

A) Zona de identificación B) Zona de declaración de datos
C) Zona de declaración de sentencias D) Zona de declaración de programas
E) Zona de declaración de funciones

2. De los siguientes grupos de "palabras", identifica a las tres que son identificadores

A) 12rojo, robot, salario
B) robot, inventario, repeat
C) repeat, while, procedure
D) procedimiento, funcion, programa
E) procedimiento, robot, procedure

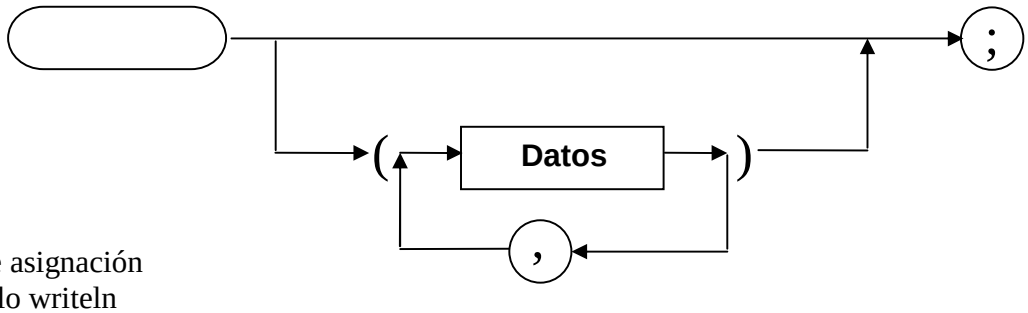
3. De los siguientes grupos, identifica a las tres que son palabras reservadas

A) robot, salario, programa
B) robot, inventario, repeat
C) repeat, program, procedure
D) procedimiento, funcion, programa
E) procedimiento, robot, procedure

4. Son datos de tipo primitivo:

A) solo carácter, lógico y real
B) solo carácter, real y entero
C) booleano, carácter, real y entero
D) lógico booleano, real y entero
E) solo booleano, lógico y real.

5. El diagrama sintáctico siguiente, corresponde a las sentencias:



A) De asignación
B) Solo writeln
C) Solo readln
D) If-then-else
E) writeln y readln

Bibliografia.

Joyanes , Aguilar Luis, Programación en Turbo Pascal versión 5.5, 6.0 y 7.0.
Editorial McGraw-Hill.
Stephen K. O´Brien, Steve Nameroff. Turbo Pascal 7.
Editorial McGraw-Hill.
Leobardo López R. Programación Estructurada en Turbo Pascal 7.
Editorial Alfaomega.
Mat. Gilberto Fuentes Romero. Cibernética y computación II, Lenguaje de Programación Pascal. C.C.H. Plantel Sur.

UNIDAD II. ESTRUCTURAS DE CONTROL DE SECUENCIAS

Introducción

En los lenguajes de programación, las **estructuras de control** te permiten modificar el flujo de ejecución de las instrucciones de un programa mediante las sentencias condicionales y de ciclo.

Con las estructuras de control se puede:

- Ejecutar un grupo u otro de sentencias (If-Then-Else y Case), dependiendo de una condición
- Ejecutar un grupo de sentencias **mientras** exista una condición (While-do)
- Ejecutar un grupo de sentencias **hasta** que exista una condición (Repeat-Until)
- Ejecutar un grupo de sentencias un número determinado de veces (For-do)

Todas las estructuras de control tienen un único punto de entrada y un único punto de salida. Esto es una de las cosas que permiten que la programación se rija por los principios de la programación estructurada.

Los lenguajes de programación modernos tienen estructuras de control similares.

Básicamente lo que varía entre las estructuras de control de los diferentes lenguajes es su sintaxis, cada lenguaje tiene una sintaxis propia para expresar la su estructura.

Para alcanzar el propósito y los aprendizajes especificados, realiza investigaciones sobre la clasificación de las estructuras de control, así como de las sentencias simples y compuestas cuyo fin es que apliquen estos conocimientos en el desarrollo de programas que resuelven problemas que involucran el uso de dichas estructuras de control. De igual forma retoma tus conocimientos previos sobre los diversos tipos de operadores.

Los problemas que se abordan requieren las estructuras tanto incondicionales como las condicionales (selectivas y/o de ciclo).
Para mayores referencias, consultar la Guía de Cibernética y Computación I Unidad III y Unidad IV, y la Unidad I de Cibernética y Computación II.

Propósito

Al finalizar la unidad utilizarás las sentencias de condición, selección y ciclo, en la construcción de programas para resolver problemas en el lenguaje de programación.

Aprendizajes a lograr:

- Identificarás que el orden de ejecución de sentencias es de arriba hacia abajo (Top-Down)
- Utilizarás como estructura condicional de selección simple IF-THEN Y DOBLE IF-THEN-ELSE y como un caso particular CASE.
- Emplearás las estructuras de ciclo: WHILE-DO, FOR-DO y REPEAT-UNTIL.
- Explicarás las diferencias entre las estructuras de control.
- Describirás la sintaxis y semántica de las estructuras de control.
- Elaborarás programas que involucran las estructuras de control.

Estrategias de aprendizaje:

- Describirás las características de las estructuras de control.
- Enfatizarás la importancia de las expresiones lógicas en el funcionamiento de las sentencias condicionales.
- Enfatizarás la conveniencia del uso del tabulador para dentar los programas como una ayuda visual para la identificación de sentencias compuestas.
- Ejemplificarás el uso de las estructuras de control por medio de programas ya elaborados.
- Resolverás y utilizar un mismo problema en el que se puedan mostrar cada una de las diferentes estructuras.
- Realizarás la prueba de escritorio de los diagrama de flujo o pseudocódigo.
- Utilizarás los problemas desarrollados en la unidad tres y cuatro de la “Guía de Cibernética y Computación I y la unidad I de la Guía de Cibernética y Computación II”.

Diseño Top-Down

La metodología descendente (Top-Down), también conocida como arriba-abajo es establecer una serie de niveles de menor o mayor complejidad (arriba-abajo) que den solución al problema. Consiste en efectuar una relación entre las etapas de la estructuración de forma que una etapa jerárquica y su inmediatamente inferior se relacionen mediante entradas y salidas de información.

Recuerda identar tus sentencias y realizar los comentarios pertinentes para que el programa sea fácil de leer y entender aún para las personas que no lo realizaron.

Contadores y Acumuladores

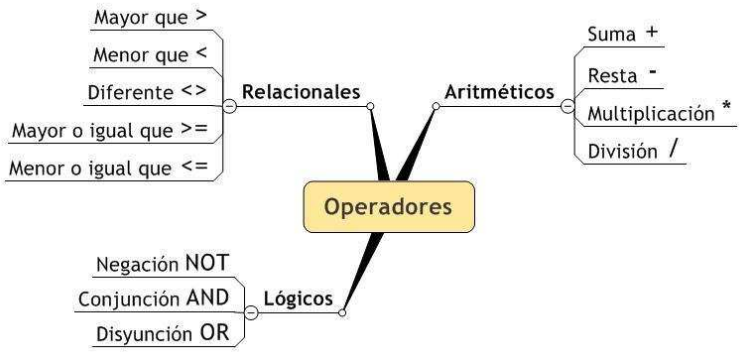
Definición: Un **contador**, es una variable que se incrementa en la unidad o en una cantidad constante.
Sintaxis: Contador := Contador + 1;
Ejemplo: En las carreras automovilísticas de Fórmula 1, el contador informa al piloto el número de de vueltas que lleva el piloto en la carrera.

Definición: Un **acumulador** es una variable que se incrementa en una cantidad variable.
Sintaxis: Acumulador:= Acumulador + X; { x variable}
Ejemplo: En una cuenta bancaria, depositas tu dinero mensualmente, la cantidad que depositas seria la variable y el interés también, cuando consultas el saldo el acumulador te ayuda a sumar estas cantidades.

Las condiciones son expresiones lógicas y se construyen con operadores relacionales y lógicos.

Utilización de operadores

Para resolver problemas apoyándote en la computadora es conveniente recordar que existen varios operadores que están involucrados en el uso de las estructuras de control, por lo que te proponemos este mapa mental para tener es forma esquemática un panorama de ellos.



Definición: Una **bandera**, también denominada interruptor o conmutador es una variable que puede tomar uno de dos valores (verdadero o falso) a lo largo de la ejecución del programa y permite comunicar información de una parte a otra del mismo.

El tipo de dato primitivo boolean del lenguaje de programación Pascal es una expresión lógica que puede tomar un valor de dos posibles verdadero o falso. Por omisión el valor es falso.

Sentencias compuestas

Las sentencias compuestas son grupos de sentencias, separadas cada una por un punto y coma ";" que son tratadas como una sola sentencia.

Para identificar una sentencia compuesta de un grupo sucesivo de sentencias se encierran entre las palabras reservadas **BEGIN** y **END**. Uno de los ejemplos más claros de una sentencia compuesta es el cuerpo de un programa principal en Pascal, el lenguaje toma todo lo que existe entre estas dos palabras como un solo elemento a ejecutarse aún cuando contenga varias sentencias:

```
PROGRAM Prueba;
BEGIN
    WriteLn('Primera línea de una sentencia compuesta');
    WriteLn('Segunda línea de una sentencia compuesta');
    WriteLn('Tercera línea de una sentencia compuesta');
END.
```

El punto y coma que se encuentra antes de la palabra reservada **END** puede ser suprimido sin afectar a la compilación.

En programación es frecuente la aplicación de sentencias condicionales en la solución de problemas. Para que los educandos comprendan su sintaxis y semántica describimos a cada una de ellas y las utilizamos en la solución de problemas.

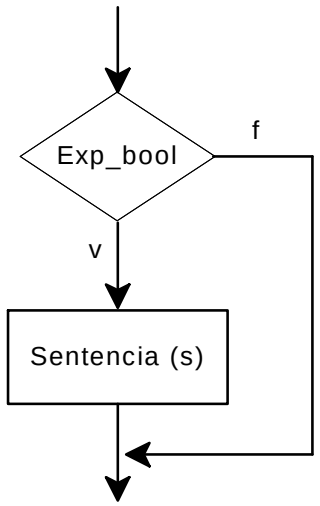
Estructuras condicionales

1. IF THEN
2. IF THEN ELSE
3. CASE OF

Para comprender su sintaxis y semántica de cada una de ellas se da el diagrama de flujo, pseudocódigo y la codificación.

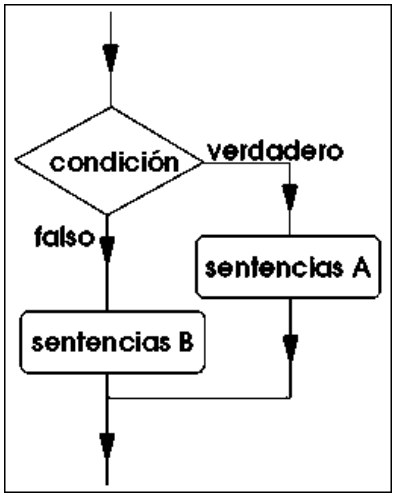
Sentencia condicional simple IF THEN

Esta sentencia es utilizada para ejecutar una sentencia en el caso de que la condición exp_bol sea verdadera.

Diagrama de flujo	Codificación
	IF exp_bool Then Sentecia(s);

Sentencia condicional doble IF THEN ELSE

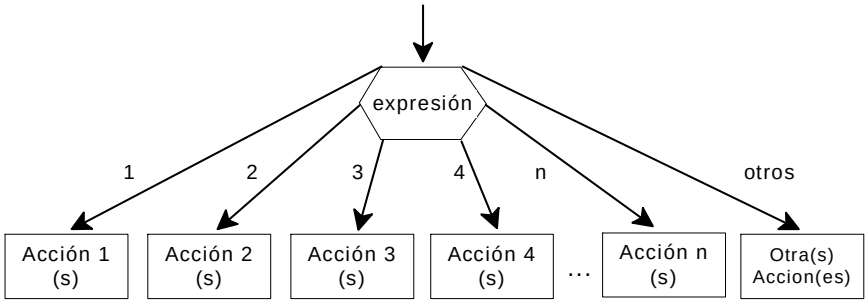
Esta sentencia se utiliza para ejecutar una sentencia en caso de que la condición sea verdadera, de lo contrario se ejecuta una sentencia distinta.

Diagrama de flujo	Codificación
	IF condicion Then Sentecias A ELSE Sentencias B;

Donde **condición** es la expresión lógica que se evaluará, cuando sea verdadera se ejecutará sentencias A, en caso contrario se ejecutará las sentencias B. En el caso que ambas sentencias sean compuestas, deberán delimitarse entre Begin y End.

Sentencia de selección CASE OF

Esta sentencia se utiliza para la elección de una alternativa de entre varias posibles, para ello, primero se compara el valor de expresión con la lista de casos, si la comparación resulta ser 1, se ejecuta acción 1, si resulta ser 2, se ejecuta acción 2, ..., si resulta ser n, se ejecuta acción n, de otro modo, se ejecuta otra(s) acciones. En caso que las sentencias sean compuestas, deben delimitarse por Begin y End.

Diagrama de flujo	Codificación
	**Case expresion OF** 1: accion 1(s) 2: accion 2(s) 3: accion 3(s) 4: accion 4(s) . . . . . . n: accion n(s) Else otra(s) acciones; end;

Para utilizar las sentencias condicionales en la solución de problemas con el apoyo de la computadora y el lenguaje de programación Pascal, retomaremos los algoritmos desarrollados en la tercera unidad de la guía de Cibernética y Computación I, para la solución de los problemas de salario neto, días y deporte.

Problema salario neto. Encuentra el salario semanal neto (salneto) que la empresa "Terabyte", pagará a uno de sus empleados considerando el nombre del empleado (nombre), las horas trabajadas a la semana (hor_trab), la tarifa por hora (tar_hora) y la tasa de impuestos de acuerdo a las siguientes categorías (cat), para la categoría A el impuesto es del 5% (imp_a), para la categoría B el impuesto es del 10% (imp_b) y para la categoría C el impuesto es del 15% (imp_c). Deberá visualizarse como salida el nombre del empleado, las horas trabajadas, la tarifa por hora, la categoría, el salario bruto y el salario neto.

Actividad de aprendizaje. Escribe en la tabla los datos que completan el análisis del problema.

Datos entrada	Tipo	Relaciones (cómputos)	Datos salida	Tipo
nombre	cadena	*salbruto* ←		
		salneto ←		
tar_hor	entero			
cat	caracter			

Solución del problema con la sentencia condicional simple IF THEN

Pseudocódigo	Codificación
Actividad de aprendizaje. Escribe en las rayas, las sentencias faltantes en el pseudocódigo. Inicio leer nombre leer hor_trab _____ _____ salbruto ⟵ hor_trab*tar_hora si (cat='A') o (cat='a') entonces salneto ⟵ salbruto-salbruto*imp_a fin_si _____ _ _____ _ _____ _ Si (cat='C ') o (cat='c') entonces Salneto ⟵ salbruto-salbruto*imp_c fin_si escribir nombre escribir hor_trab escribir tar_hora _____ _____ _____ Fin	**Actividad de aprendizaje**. Escribe en las rayas, las líneas de código faltantes. `program salario_neto;` `uses crt;` `const` `imp_a=0.05;` `imp_b=0.10;` `imp_c=0.15;` `var` `nombre:string[10];` `hor_trab,tar_hora:integer;` `salbruto:integer;` `salneto:real;` `cat:char;` `Begin` `write('Digita nombre :');` `readln(nombre);` `write('Introduce horas :');` `readln(hor_trab);` `write('Digita tarifa por hora :');` `readln(tar_hora);` `write('Digita categoria :');` `readln(cat);` `salbruto:=hor_trab*tar_hora;` `if(cat='a') or (cat='A') then` `salneto:=salbruto-salbruto*imp_a;` _____ _ _____ _ _____ _ _____ _ `writeln(nombre);` `writeln(hor_trab);` `writeln(tar_hora);` `writeln(cat);` `writeln(salbruto);` `writeln(salneto:4:2);` `readln;` `end.`

Solución del problema con la sentencia condicional IF THEN ELSE

Pseudocódigo	Codificación
Actividad de aprendizaje. Escribe en las rayas las sentencias faltantes en el pseudocódigo. Inicio leer nombre leer hor_trab _____ _____ salbruto ⟵ hor_trab*tar_hora si (cat='A') o (cat='a') entonces salneto ⟵ salbruto-salbruto*imp_a sino _____ _____ _____ Si (cat='C') o (cat='c') entonces salneto ⟵ salbruto-salbruto*imp_c sino escribir ('Opción invalida') escribir nombre escribir hor_trab escribir tar_hora _____ _____ _____ Fin	**Actividad de aprendizaje**. Escribe en las rayas las sentencias faltantes en el código. `program salario_neto;` `uses crt;` `const` `imp_a=0.05;` `imp_b=0.07;` `imp_c=0.10;` `var` `nombre:string[10];` `hor_trab,tar_hora,salbruto:integer;` `salneto:real;` `categoria,sigue:char;` `Begin` `write('Digita nombre :');` `readln(nombre);` `write('Introduce horas :');` `readln(hor_trab);` `write('Digita tarifa por hora :');` `readln(tar_hora);` `write('Digita categoria :');` `readln(cat);` `salbruto:=hor_trab*tar_hora;` `if(cat='a') or (cat='A') then` `salneto:=salbruto-salbruto*imp_a` `else` _____ _____ _____ `if (cat='c')or (cat='C') then` `salneto:=salbruto-salbruto*imp_c` `else` `writeln('Opción invalida');` `writeln(nombre);` `writeln(hor_trab);` `writeln(tar_hora);` `writeln(cat);` `writeln(salbruto);` `writeln(salneto:4:2);` `readln;` `end.`

Solución del problema con la sentencia de selección CASE OF

Pseudocódigo	Codificación
Actividad de aprendizaje. Escribe en las rayas las sentencias faltantes en el pseudocódigo. Inicio leer nombre leer hor_trab _____ _____ salbruto ⟵ hor_trab*tar_hora según_sea cat hacer 'A','a': salneto ⟵ salbruto-salbruto*imp_a _____ _____ En caso contrario Escribir ('Opcion invalida') Fin_caso escribir nombre escribir hor_trab escribir tar_hora _____ _____ _____ Fin	**Actividad de aprendizaje**. Escribe en las rayas, las sentencias del diagrama de flujo, faltantes en el pseudocódigo. `program salario_neto;` `uses crt;` `const` `imp_a=0.05;` `imp_b=0.07;` `imp_c=0.10;` `var` `nombre:string[10];` `hor_trab,tar_hora,salbruto:integer;` `salneto:real;` `cat:char;` `Begin` `write('Digita nombre :');` `readln(nombre);` `write('Introduce horas :');` `readln(hor_trab);` `write('Digita tarifa por hora :');` `readln(tar_hora);` `write('Digita categoria :');` `readln(cat);` `salbruto:=hor_trab*tar_hora;` `case cat of` `'a','A': salneto:=salbruto-salbruto*imp_a` `_____` `_____` `else` `writeln('Opción invalida');` `end;` `writeln(nombre);` `writeln(hor_trab);` `writeln(tar_hora);` `writeln(cat);` `writeln(salbruto);` `writeln(salneto:4:2);` `readln;` `end.`

Problema deporte. Selecciona el deporte que es apropiado practicar, considerando como dato la temperatura (temp) en grados Fahrenheit con base en la siguiente tabla:

Deporte	Temperatura
Natación	temp mayor que 85
Tenis	temp mayor que 70, pero menor o igual a 85.
Golf	temp mayor que 32, pero menor que 70.
Esquí	temp mayor que 10, pero menor o igual a 32.
Marcha	temp menor o igual que 10

Actividad de aprendizaje. Escribe en la tabla los datos que completan el análisis del problema.

Datos entrada	Tipo	Relaciones (cómputos)	Datos salida	Tipo

Pseudocódigo	**Codificación**
Actividad de aprendizaje. Escribe en las rayas las sentencias faltantes en el pseudocódigo. Inicio Leer temp Si (temp > 85) entonces Escribir (“Natación”) Sino _____ _____ _____ Si (temp > 32) entonces Escribir(“Golf”) Sino _____ _____ _____ _____ Fin_temp>10 Fin_temp>32 Fin_temp>70 Fin_temp>85 Fin	**Actividad de aprendizaje**. Escribe en las rayas las sentencias faltantes en el programa deporte. program deporte; uses crt; var temp:real; begin clrscr; write('Digita temperatura: '); readln(temp); if (temp > 85) then writeln('Deporte practicado: ','Natación') else if (temp > 70) then writeln('Deporte practicado: ','Tenis'); else if (temp > 32) then writeln('Deporte practicado: ','Golf'); else _____ _____ _____ _____ readln; end.

Problema días. Escribir en pantalla el nombre del día de la semana, considerando a la variable entera día (dia).

Análisis del problema días.

Actividad de aprendizaje. Escribe en la tabla los datos que completan el análisis del problema.

Datos entrada	Tipo	Relaciones (cómputos)	Datos salida	Tipo

1. Resolución del problema con la sentencia if then

Completa el Pseudocódigo y Codificación del programa **DÍA_SEMANA**, utilizando la sentencia simple if then.

Pseudocódigo del programa dia_sem1	**Codificación del programa dia_sem1**
Actividad de aprendizaje. Escribe en las rayas las sentencias faltantes en el pseudocódigo. **Inicio** **leer** dia **si** dia=1 **entonces** **escribir**('1:Lunes') **fin_si** _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____	**Actividad de aprendizaje**. Escribe en las rayas las sentencias faltantes en el programa deporte. Program dia_sem1; uses crt; var dia:integer; **begin** clrscr; write('Digita día de la semana: '); **readln**(dia); **if** dia=1 **then** **writeln**('1:Lunes'); **end**; _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____

___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ **fin**	_____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ **end.**

2. Resolución del problema con la sentencia if then else

Completa el Pseudocódigo y Codificación del programa **DÍA_SEMANA**, utilizando la sentencia doble if then else

Pseudocódigo del programa dia_semana	**Codificación del programa dia_semana**
Actividad de aprendizaje. Escribe en las rayas las sentencias faltantes en el pseudocódigo. **Inicio** **leer** dia **si** dia=1 **entonces** **escribir**('1:Lunes') **sino** **si** dia=2 **entonces** **escribir**('2:Martes')	**Actividad de aprendizaje**. Escribe en las rayas las sentencias faltantes en el programa deporte. Program dia_semana; uses crt; var dia:integer; **begin** clrscr; **if** dia=1 **then**

sino **si** dia=3 **entonces** **escribir**(‘3:Miércoles’) **sino** ____ **fin.**	**writeln**(‘1:Lunes’) **else** **if** dia=2 **then** **writeln**(‘2:Martes’) **else** ____ **End.**

3. Resolución del problema con la sentencia case of

Completa el Pseudocódigo y Codificación del programa **DÍA_SEMANA_2**, utilizando la sentencia múltiple **case of**

Pseudocódigo del programa dia_semana_2	Codificación del programa dia_semana_2

Inicio **escribir**(‘Digita número de día de la Semana: ‘) **leer** dia **según_sea** dia **hacer** 1:**escribir**(‘1:Lunes’) 2:**escribir**(‘2:Martes’) _____ ___ **Fin.**	Program dia_semana_2; uses crt; var dia:integer; **begin** clrscr; **write**(‘Digita día de la semana’); **readln**(dia) **case** dia **of** 1:**writeln**(‘1:Lunes’); 2:**writeln**(‘2:Martes’); _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ _____ ___ End.

Los programas que desarrollaste. Edítalos, compílalos y ejecútalos varias veces, para que te familiarices con el ambiente integrado del compilador de Pascal y con el uso de la computadora en la solución de problemas. Si tienes errores de compilación, corrígelos, si persisten consulta a un profesor de Cibernética y Computación.

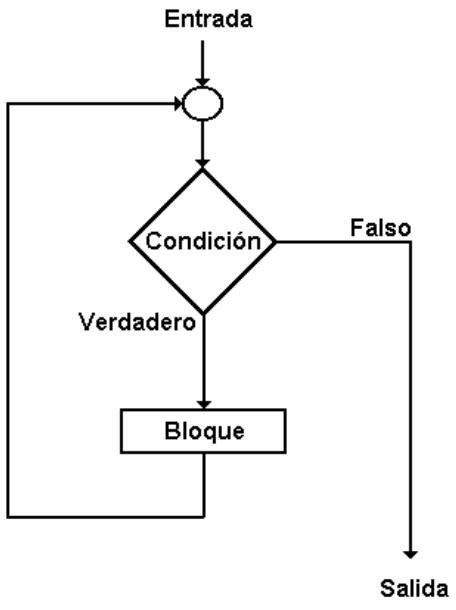
Estructuras de ciclo

Las sentencias de ciclo se utilizan para la repetición de una sentencia o de un conjunto de sentencias, dependiendo de la variable lógica que controla al ciclo. El lenguaje de programación Pascal soporta a las siguientes:

1. WHILE DO
2. REPEAT UNTIL
3. FOR DO

Ciclo WHILE

El ciclo WHILE ofrecen la ventaja de que la ejecución se realiza mientras se cumpla una condición, por lo tanto es posible controlar el número de repeticiones una vez iniciado el ciclo. Para familiarizar a los educandos con ella, damos su diagrama de flujo y la codificación para comprender su sintaxis y semántica.

Diagrama de flujo	Codificación
	WHILE condición DO Bloque;

Donde condición es la expresión booleana que controla al ciclo. El ciclo se ejecutará mientras que la condición sea verdadera, en caso contrario, el ciclo no

se ejecutará. Cuando la sentencia contenida en Bloque sea compuesta deberá delimitarse entre Begin y End.

Un programa que escriba los números del 1 al 50, utilizando el ciclo WHILE se vería como sigue:

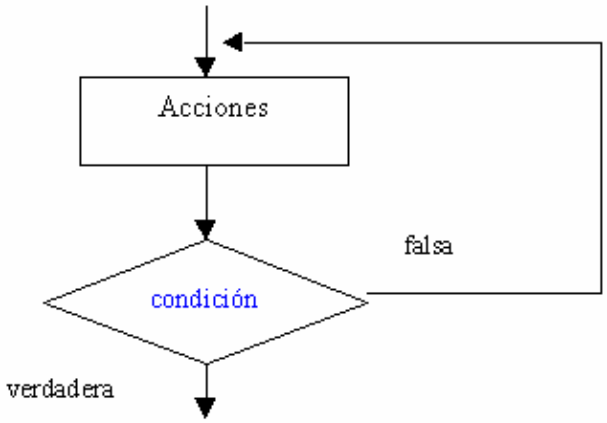
```
PROGRAM Ciclo_WHILE;
VAR
   Numero: Integer;

BEGIN
      Numero:= 1;
      WHILE (Numero <= 50) DO
         BEGIN
               WriteLn (Numero);
               Numero:= Numero +1;
         END;
END.
```

Al final del programa la variable Numero guardará el valor 51, siendo el valor que no cumplió con la condición establecida en el ciclo WHILE.

Ciclo REPEAT-UNTIL

El ciclo REPEAT es muy parecido al ciclo WHILE, la diferencia entre ambos es que en WHILE la condición se evalúa al principio del ciclo, en cambio en REPEAT se evalúa al final, lo que significa que en un ciclo REPEAT la sentencia se ejecutará por lo menos una vez y con el ciclo WHILE puede ser que no se ejecute ninguna vez. Para familiarizar a los educandos con la sentencia repeat, damos su diagrama de flujo y codificación.

Diagrama de flujo	Sintaxis
	REPEAT Acciones; UNTIL condicion

Donde condición es la expresión booleana que controla al ciclo REPEAT. El ciclo se ejecutará cuando la condición sea falsa, en caso contrario, el ciclo terminará.

Ejemplo: el programa para la escritura de los cincuenta números lo desarrollamos con el ciclo REPEAT y es como sigue:

```
PROGRAM Ciclo_RepeatUntil;
VAR
   Numero : Integer;
BEGIN
  Numero := 1;
  REPEAT
       WriteLn (Numero);
      Numero := Numero + 1;
  UNTIL Numero > 50;
END.
```

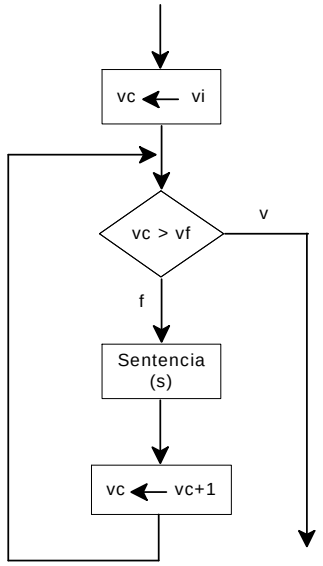
En la resolución de problemas es necesario adecuarlo, para que el programa sea aplicable en un entorno más generalizado y no solo para un problema específico.

Ciclo FOR

El ciclo **FOR** repite una sentencia un determinado número de veces que se indica al momento de llamar al ciclo. Incrementa una variable en uno desde un valor inicial hasta un valor final ejecutando en cada incremento la sentencia que se quiere repetir.

Su sintaxis es: FOR identificador := inicio TO fin DO instrucción;

Identificador: variable que se incrementará, **inicio:** primer valor que tendrá dicha variable y **fin**: es el valor hasta el cual se incrementará la misma; **instrucción**: es la sentencia (sencilla o compuesta) que se ejecutará en cada incremento de la variable.

Diagrama de flujo	Sintaxis
	FOR ASCENDENTE FOR vc:=vi TO vf do Sentencia(s); FOR DESCENDENTE FOR vc:=vf DOWNTO vi do Sentencia(s);

El siguiente ejemplo escribe los números del 1 al 50 en pantalla. La variable utilizada es "Numero".

```
PROGRAM Ciclo_FOR;
VAR
Numero : Integer;

BEGIN
  FOR Numero := 1 to 50 DO
  WriteLn(Numero);
END.
```

Una de las limitaciones de los ciclos FOR es que una vez iniciado el ciclo se ejecutará el número de veces predefinido sin posibilidad de agregar o eliminar ciclos en su ejecución.

Es posible hacer que un ciclo FOR cuente hacia atrás (descendente), es decir que la variable decremente. Para esto cambiamos la palabra **TO** por **DOWNTO**, y colocamos el valor mayor a la izquierda y el menor a la derecha. Ejemplo:

```
PROGRAM Ciclo_FOR_2;
VAR
Numero : Integer;
BEGIN
```

```
FOR Numero := 50 DOWNTO 1 DO
WriteLn(Numero);
END.
```

En ambos casos para la sentencia de ciclo FOR ascendente y descendente, la variable de control del ciclo se le asigna el valor inicial, la cual es incrementada de manera automática por la sentencia FOR. En cambio, en las sentencias WHILE y REPEAT es necesaria la sentencia que incremente el valor a la variable de control del ciclo.

Resolución del problema utilizando la computadora

Actividad de aprendizaje. Revisa la codificación del programa salario_neto.pas en el Disco anexo utilizando las sentencias de ciclo *REPEAT-UNTIL* y de selección *IF-THEN-ELSE*, corrige los errores cometidos.

Actividad de aprendizaje. Revisa la codificación del programa salario_neto2.pas en el Disco anexo utilizando las sentencias de ciclo *WHILE-DO* y de selección *IF-THEN-ELSE*, corrige los errores.

Actividad de aprendizaje. Revisa la codificación del programa salario_neto3.pas en el Disco anexo utilizando las sentencias de ciclo *REPEAT-UNTIL* y de selección *CASE-OF*, corrige los errores cometidos.

Actividad de aprendizaje. Revisa la codificación del programa salario_neto4 en el Disco anexo utilizando las sentencias de ciclo *WHILE-DO* y de selección *CASE OF*, corrige los errores cometidos.

Nota: El Disco anexo contiene otros programas, revísalos e indica que es lo que hace cada uno de ellos, realiza la prueba de escritorio y en caso necesario corrígelos.

ACTIVIDADES DE RETROALIMENTACIÓN

Elabora el programa para resolver los siguientes problemas:

PROBLEMA 1. (DÍA_SEMANA): Realiza la adecuación al programa dia_semana con las sentencias de ciclo while y repeat, para que se ejecute para varias corridas.

PROBLEMA 2. (DEPORTE): Realiza la adecuación al programa deporte con la sentencia condicional simple if then.

PROBLEMA 3. (DEPORTE): Realiza la adecuación al programa deporte con las sentencias de ciclo while y repeat, para que la ejecución sea para varias corridas.

PROBLEMA 4. (SALARIO BRUTO): Realiza la adecuación al programa salario neto con las sentencias de ciclo while y repeat, para que la ejecución sea para varias corridas.

5. ¿Cuáles son las diferencias entre las sentencia de ciclo?
While do _____

_____

Repeat until

_____

_____

For do

_____

_____

6. Problema (PROMEDIO). Encuentra el promedio general de aprovechamiento de los alumnos del grupo 663 de Cibernética y Computación II, considerando 4 calificaciones por alumno, cuando el promedio sea al menos seis deberá aparecer la leyenda “Aprobado”, en caso contrario la leyenda “Reprobado”. Los nombres y calificaciones serán introducidas desde el teclado y los datos de salida serán como siguientes:

NOMBRE	CAL1	CAL2	CAL3	CAL4	PROMEDIO
PROMEDIO GENERAL DEL GRUPO:					

7. Responde las siguientes preguntas:

1. ¿Cuál es el orden de ejecución de un programa en: Free Pascal?
   _____
   _____
2. ¿Qué se entiende por estructuras incondicionales?
   _____
   _____
3. ¿Qué es una condición lógica?
   _____
   _____
4. Menciona cuantos tipos de operadores hay:
   _____
   _____
5. ¿Qué es un bucle?
   _____
   _____
6. ¿Qué es una iteración?
   _____
   _____
7. ¿Qué se entiende por estructuras condicionales?
   _____
   _____

8. ¿Cómo funciona la estructura IF - THEN e IF – THEN - ELSE?
_____
_____

9. ¿Cómo funciona la estructura WHILE - DO?
_____
_____

10. ¿Cómo funciona la estructura CASE - OF?
_____
_____

11. ¿Cómo funciona la estructura REPEAT - UNTIL?
_____
_____

12. ¿Cómo funciona la estructura FOR – DO?
_____
_____

13. ¿Qué es una bifurcación?
_____
_____

Reactivos muestra para el examen extraordinario.

1. La técnica que permite establecer el orden en la ejecución de sentencias corresponde _____ para la solución de un problema.

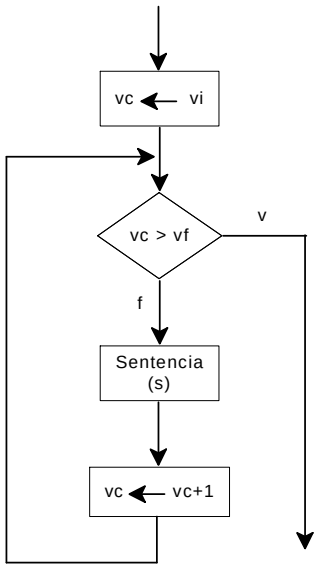
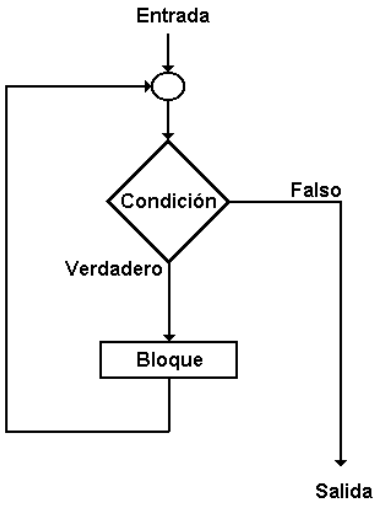
a) a la secuencial
b) a la modular
c) de arriba hacia abajo
d) al refinamiento sucesivo
e) al diseño descendente

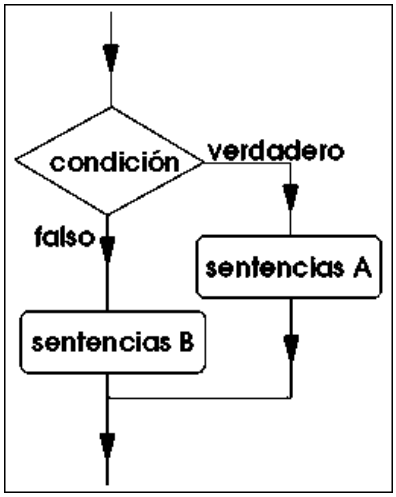
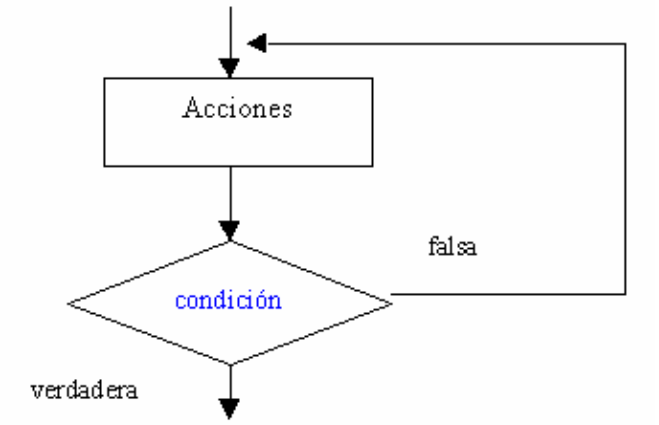
2. Elige la sentencia de control con su descripción

Sentencias de control	Descripción
a. IF THEN ELSE	I. Sentencia de ciclo que se ejecuta hasta que la variable de control es mayor que la variable final.
b. FOR DO	II. Sentencia de ciclo que se ejecuta mientras que la condición de ciclo sea verdadera.
c. REPEAT UNTIL	III. Sentencia condicional que ejecuta una acción cuando la condición es verdadera, en caso contrario ejecuta una acción alterna.
d. WHILE DO	IV. Sentencia de ciclo que se ejecuta hasta que la condición de ciclo sea verdadera.

A) a - III, b - I, c - IV, d - II
B) a - II, b - III, c - I, d - IV
C) a - III, b - I, c - II, d - IV
D) a - IV, b - II, c - III, d - I
A) a - I, b - IV, c - II, d - III

3. Elige la sentencia de control con su sintaxis

Sentencia de control	Sintaxis
a.  	I. REPEAT ACCION UNTIL EXP_BOOL
b.  	II. IF EXP_BOOL THEN ACCION_A ELSE ACCION_B;

c.  	III. WHILE (EXP_BOOL) DO ACCION;
d.  	IV. FOR VC:=VI TO VF DO ACCION;

A) a - III, b - IV, c - I, d - II
B) a - IV, b - III, c - II, d - I
C) a - II, b - I, c - IV, d - III
D) a - IV, b - III, c - I, d - II
A) a - I, b - II, c - III, d - IV

4. ¿Cuáles son las sentencias del programa deporte, para que se ejecuta para varias corridas?

```
program deporte;
  uses crt;
  var
     temp:real;
     _____________________      1
  begin
       _____________________________  2
       _____________________________  3
       begin
          clrscr;
          write('Digita temperatura: ');
          readln(temp);
          if (temp > 85 ) then
            writeln('Deporte practicado: ','Natación')
          else
           if (temp > 70 ) then
             writeln('Deporte practicado: ','Tenis');
          else
             if (temp > 32 ) then
               writeln('Deporte practicado: ','Golf');
            else
               if (temp > 10)  then
                 writeln('Deporte practicado: ','Golf')
               else
                  writeln('Deporte practicado:
','Marcha');
          write('Otra corrida S/N: ');
          _____________________________   4
          readln;
       end;
end.
```

A) readln(sigue);
while (sigue=’s’ or sigue=’S’) do
sigue: char;
sigue:= ‘s’;

B) sigue: char;
while (sigue=’s’ or sigue=’S’) do
readln(sigue);
sigue:= ‘s’;

C) sigue: char;
readln(sigue);
sigue:= ‘s’;
while (sigue=’s’ or sigue=’S’) do

D) sigue: char;
sigue:=’s’;
while (sigue=’s’ or sigue=’S’) do
readln(sigue);

E) while (sigue=’s’ or sigue=’S’) do
readln(sigue)
sigue: char;
sigue:= ‘s’;

UNIDAD III. PROCEDIMIENTOS Y FUNCIONES

Introducción

Una estrategia para la resolución de problemas complejos con computadora, es la división o descomposición del problema en otros problemas más pequeños ó subproblemas. Estos subproblemas se implementan (se realizan) mediante módulos o subprogramas. Los subprogramas son una herramienta importante para el desarrollo de algoritmos y programas, de modo que normalmente un proyecto de programación, se compone generalmente, de un programa principal y un conjunto de subprogramas o módulos, con las llamadas a los mismos dentro del programa principal. Los subprogramas se clasifican en **procedimientos y funciones.**

Propósito

Al finalizar esta unidad, utilizarás los procedimientos y funciones, para elaborar programas de estructura modular, mediante el desarrollo de programas aplicados a la solución de problemas.

Aprendizaje

Al finalizar esta unidad aprenderás:

- Ha comprender la importancia de subdividir un problema en subproblemas en la cual aplicarás la programación modular.
- Comprenderás la lógica de la programación modular que se basa en que resulta más fácil escribir un buen programa si se divide en partes más pequeñas.
- Comprenderás el concepto de programación modular, funciones, procedimientos y parámetros por valor, referencia y variable.
- Identificarás en la estructura de los programas modulares, la zona de declaración e innovación de funciones y procedimientos.
- Elaborarás algunos programas utilizando la programación modular.
- Conocerás la utilidad de manejar parámetros globales como medios de comunicación de información entre módulos y el programa principal.
- Comprenderás la diferencia entre los parámetros por valor, referencia y variable.

Estrategias de aprendizaje

- Harás una investigación documental sobre la programación modular y contestarás el cuestionario que se te pide en esta guía.
- Ejemplificarás programas sencillos donde se apliquen los procedimientos y funciones.
- En esta guía aparecen algunas ayudas como son los análisis de los problemas que se presentan, para que codifiques los programas en forma estructurada.
- Emplearás los procedimientos y funciones que te permitan codificar y ejecutar los programas que aparecen en esta guía.
- En los ejemplos y ejercicios para esta unidad realizarás desde la etapa de análisis y diseño, para enfatizar la importancia de construir los programas como colección de tareas genéricas relacionadas entre si y no como una lista de instrucciones individuales.
- Durante el desarrollo de los programas emplearás la secuencia:
  a) Definición de la estructura del cuerpo principal, señalando encabezados del programa principal, señalando encabezados de procedimientos y funciones involucradas, así como los parámetros requeridos.
  b) Declaración de variables globales.
  c) Llamados de las funciones y procedimientos dentro del programa principal.
- Señalar las ventajas y limitaciones en el uso de las funciones en relación con los procedimientos.

Actividades de aprendizaje

Programación modular con procedimientos y funciones.

Contesta las siguientes preguntas.

¿Qué significa programación modular? _____
_____

¿Qué es un módulo? _____
_____

¿Cuándo es útil la modularización? _____
_____

¿Cuáles son las ventajas de la modularización de programas? _____
_____

¿Qué tipo de módulos hay disponibles en Pascal? _____
_____

¿Qué es un procedimiento? _____
_____

¿Cómo se declara un procedimiento? _____
_____

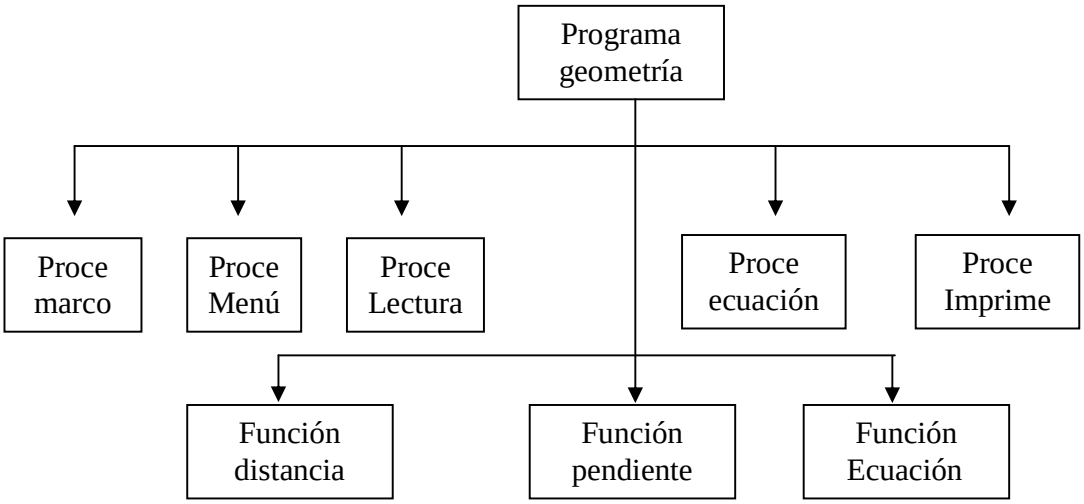
¿Cómo se llama a un procedimiento? _____
_____

¿Cuáles son las partes principales que conforman un procedimiento? _____
_____
¿Cuál es la diferencia entre variables locales y variables globales? _____
_____
¿Qué es una función? _____
_____

¿En que se diferencia una función de un procedimiento? _____
_____
¿Dónde se declara una función de un programa? _____
_____
¿Hay algún orden particular que deba de ser observado entre declaraciones de procedimiento y declaraciones de función? _____
_____

El programa geometría, te servirá para que te familiarices con los elementos de la programación modular.

Problema 1. (geometría analítica). Elabora el programa que te permita encontrar la distancia, pendiente, ecuación de la recta y el ángulo de inclinación que determinan dos puntos en el plano cartesiano XY. El diseño de este programa se muestra en el siguiente diagrama.



Actividad de aprendizaje.

Abre, compila y ejecuta el programa geometría, anexo en el CD ROM y contesta lo que se te pide:

¿Cuántos módulos tiene el programa? _____

_____

¿Cuáles son funciones? _____

_____

¿Cuáles son procedimientos? _____

_____

Describe al procedimiento. _____

_____

Describe la función. _____

_____

Identifica las variables globales. _____

_____

Identifica las variables locales en cada uno de los módulos._____

_____

Identifica los parámetros valor en los módulos. _____

_____

Identifica los parámetros variables. _____

_____

Escribe la declaración de cada uno de los módulos. _____

_____

¿Cómo se ejecuta a cada uno de los módulos? _____

_____

Actividad de aprendizaje. Describe la acción de

Del procedimiento marco _____

_____

Del procedimiento menú_____

_____

Del procedimiento lectura_____

_____

De la función distancia _____

_____

De la función pendiente _____

_____

De la función ecuación _____

_____

Del procedimiento imprime _____

_____

A continuación se te da parte de la codificación del programa áreas, para determinar el perímetro y área de un rectángulo mediante una función y un procedimiento, con base en la codificación de la función y procedimiento, codifica la función para calcular el área del cilindro y el procedimiento para calcular su volumen; y la función para calcular el área de una esfera y el procedimiento para calcular su volumen.

```
program areas;
uses crt;
var
  base,altura,radio:integer;
  opcion:integer;

  procedure marco;
    const
       renini=4;
       colini=4;
       renfin=23;
       colfin=70;
    var
     col,ren,i:integer;
    begin
       textcolor(red);
       clrscr;
       gotoxy(colini,renini);
       write('╔');
       gotoxy(colini,renfin);
       write('╚');
       gotoxy(colfin,renini);
       write('╗');
       gotoxy(colfin,renfin);
       write('╝');
       for i:=colini+1 to colfin-1 do
         begin
            textcolor(white);
            gotoxy(i,renini);
            write('═');
            gotoxy(i,renfin);
            write('═');
         end;
       for i:=colini+1 to renfin-1 do
         begin
            gotoxy(colini,i);
            write('║');
            gotoxy(colfin,i);
            write('║');
         end;
```

```
  end;

procedure menu;
  begin
     marco;
     gotoxy(15,6);
     write('C A L C U L O   D E   A R E A S');
     gotoxy(15,8);
     write('1) Perímetro rectángulo');
     gotoxy(15,10);
     write('2) Area rectángulo');
     gotoxy(15,12);
     write('3) Superficie cilindro');
     gotoxy(15,14);
     write('4) Volumen cilindro');
     gotoxy(15,16);
     write('5) Superficie esfera');
     gotoxy(15,18);
     write('6) Volumen esfera');
     gotoxy(15,20);
     write('7) Salida');
     gotoxy(15,22);
     write('Elige opción: ');
     readln(opcion);
  end;

procedure lectura;
  begin
     marco;
     case opcion of
       1,2: begin
              gotoxy(15,8);
              write('Digita base: ');
              readln(base);
              gotoxy(15,10);
              write('Digita altura: ');
              readln(altura);
            end;
       3,4: begin
              gotoxy(15,8);
              write('Digita radio: ');
              readln(radio);
              gotoxy(15,10);
              write('Digita altura: ');
              readln(altura);
            end;
       5,6: begin
```

```
            gotoxy(15,8);
            write('Digita radio: ');
            readln(radio);
          end;
    end;
  end;

function peri_rect(l,a:integer):integer;
  begin
    peri_rect:=2*(l+a);
  end;

procedure peri_area_rect(l,a:integer;var peri,area:real);
  begin
    peri  :=2*(l+a);
    area  :=l*a;
  end;
```

Actividad de aprendizaje

Investiga en un libro de geometría plana las fórmulas para calcular el área y volumen de un cilindro; el área y volumen de una esfera.

Con base en las fórmulas solicitadas, codifica las funciones y procedimientos que se indican, indicando parámetros como en function peri_rect(l,a:integer):integer; y en procedure peri_area_rect(l,a:integer;var peri,area:real);

Función area_cil

_____
_____
_____

Procedimiento vol_cil;

_____
_____
_____

Función area_esf;

_____
_____
_____

Procedimiento vol_esf;
   _____
   _____
   _____

```
procedure procesa_todo(opcion:integer);
  begin
    marco;
    case opcion of
     1: begin
          gotoxy(15,10);
          write('Perímetro rectángulo = ',peri_rect(base,altura));
        end;
     2: begin
          gotoxy(15,10);
          write('Area rectángulo = ',area_rect(base,altura));
        end;
     3: begin
          gotoxy(15,10);
          write('Area cilindro = ',area_cil(radio,altura):5:2);
        end;
     4: begin
          gotoxy(15,10);
          write('Volumen cilindro = ',vol_cil(radio,altura):5:2);
        end;
     5: begin
          gotoxy(15,10);
          write('Area esfera = ',area_esf(radio):5:2);
        end;
     6: begin
          gotoxy(15,10);
          write('Volumen esfera = ',vol_esf(radio):5:2);
        end;
    else
      begin
        gotoxy(15,10);
        write('opci¢n fuera de rango, verifica');
      end;
    end;
    readln;
  end;
```

```
begin                {Inicia programa principal}
   repeat
      menu;          {Llamada al procedimiento menú}
      lectura;       {Llamada al procedimiento lectura}
      procesa_todo(opcion); {Llama al procedimiento procesa_todo}
   until opcion = 7;
end.
```

Actividad de aprendizaje.

Realiza la codificación de funciones y procedimientos faltantes en el programa.

Edita, compila, ejecuta el programa áreas y contesta lo que te pide:

Describe los procedimientos del programa áreas: _____

_____

Describe las funciones del programa áreas: _____

_____

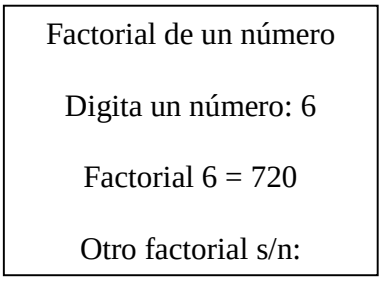
Menciona los parámetros formales: _____

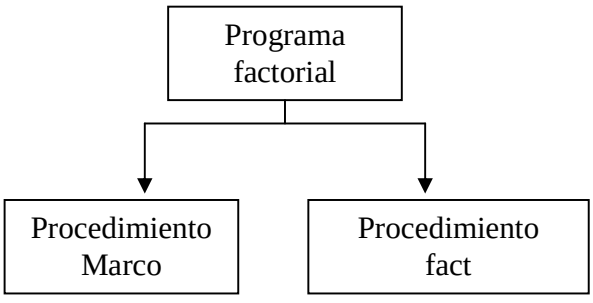
Menciona los parámetros valor: _____

Menciona los parámetros variable: _____

Menciona los parámetros actuales: _____

Problema factorial. Calcula el factorial de un número entero positivo n, tal como se especifica a continuación.





Análisis del problema: factorial:

¿Qué es el factorial del número n? _____

Describe el diagrama en las siguientes líneas en blanco. _____

_____

¿Cuál es la tarea del procedimiento marco? _____

¿Cuál es la tarea de la función fact? _____

Cabe señalar que el procedimiento marco, ya ha sido elaborado anteriormente y solo tendrás que copiarlo y adaptarlo para este problema.

Datos de entrada	tipo	datos de salida	tipo
_____	_____	_____	_____

Proceso de operaciones.

_____

Pseudocódigo: Función factorial:

```
Función Fact (i:integer): integer;
 Inicio
   SI  (i = 0) o (i = 1) entonces
        Fact  ← 1
   SINO
         Fact ← i * fact(i-1)
   Fin_si
 Fin
Fin
```

Pseudocódigo del programa principal de Factorial

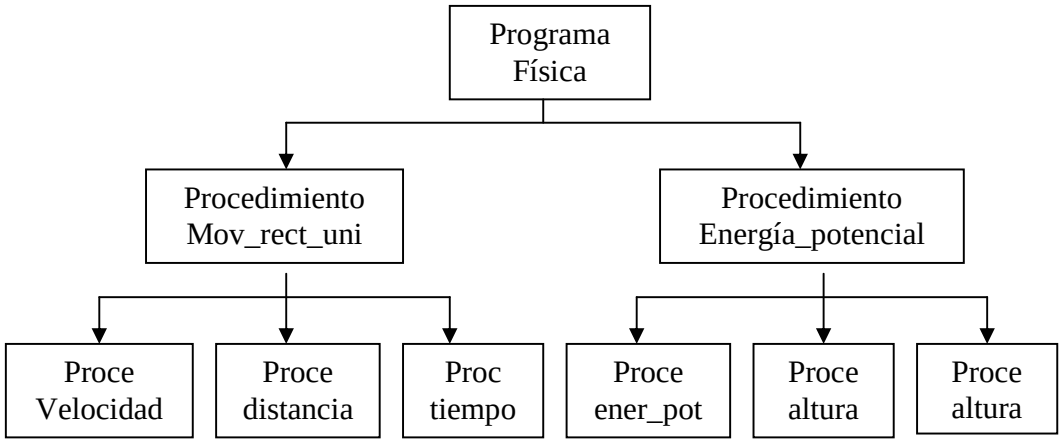
```
Inicio
  Repetir
     Marco
     Escribir ('Factorial de un número')          {Posición (10,20)}
     Escribir ('Digita número')                   {Posición (12,20)}
     Leer (num)
     Escribir ('Factorial', num,' = ', fact(num)  {Posición (14,20)}
     Escribir ('Otro Factorial s/n: ')            {Posición (16,20)}
     Leer  (sigue)
  Hasta que (sigue = 'n') o (sigue = 'N')
Fin
```

Actividad de aprendizaje.

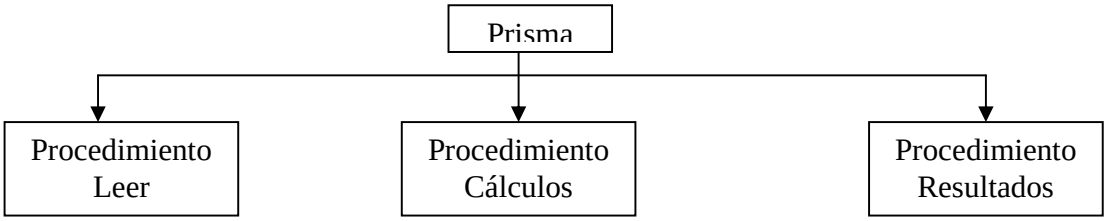
Codifica, edita, compila y ejecuta el programa factorial.

Problemas Propuestos.

1. Problema física. Calcula la velocidad, distancia y tiempo de un movimiento rectilíneo uniforme, así como la energía potencial altura y masa, como se muestra
en el diagrama Física.



2. Elabora el programa de prismas regulares, para calcular el área total tomando en cuenta el área lateral y las dos bases, también calcular el volumen. Utilizar los tres procedimientos como se muestra en el siguiente diagrama.



Algoritmo

1.- Procedimiento leer (lado, apotema, y altura).
2.- Procedimiento Calcular (lado, apotema, altura, area, vol).
Area Total = 2*bases + area lateral.
Volumen = base por altura.
3.- Procedimiento resultados (Visualiza los resultados del área y el volumen).
Declara las procedimientos en Pascal
Llama los procedimientos en el programa principal para su ejecución.

Reactivos muestra de la programación modular

1. Relaciona el concepto con su descripción.

Concepto	Descripción
a. Programación modular	I Es un módulo del programa principal que realiza una acción específica.
b. Función	II Es la descomposición del programa en módulos.
c. Procedimiento	III En un módulo del programa principal que regresa un valor a su llamada.

A) a - II, b - III, c - I
B) a - I, b - III, c - II
C) a - II, b - I, c - III
D) a - I, b - II, c - III
E) a - III, b - II, c - I

2. **Dentro de la programación modular, una función normalmente devuelve**:

A. Cero valores.
B. Un solo valor.
C. Dos valores.
D. Tres valores.
E. Más de tres valores.

3. Relaciona el parámetro con su descripción.

a. parámetro actual	I. Su valor no puede ser modificado en la ejecución del programa.
b. parámetro formal	II. Se utiliza en la llamada de la función y/o el procedimiento.
c. parámetro valor	III. Su valor puede ser modificado en la ejecución del programa.
d. parámetro variable	IV. Se declara en el encabezado del procedimiento o función.

A) a - I, b - III, c - IV, d - II
B) a - II, b - IV, c - I, d - III
C) a - III, b - IV, c - I, d - II
D) a - IV, b - III, c - II, d - I
E) a - II, b - I, c - IV, d - III

4. En el siguiente programa, ¿cuál es la variable local, el parámetro formal, el parámetro actual y la llamada al procedimiento ascii?

```
program codigo;
var
  numero:integer;
  procedure ascii(var num:integer);
  var
     caracter:char;
  begin
       write('Digita número: ');
       readln(num);
       numero:=num;
       caracter:=char(numero);
       writeln(numero,' = ',caracter);
   end;
begin
      ascii(numero);
      writeln('número = ', numero);
end.
```

a. variable local	I	ascii(numero)
b. parámetro formal	II	numero
c. parámetro actual	III	caracter
d. llamada al procedimiento	IV	num

A) a - IV, b - III, c - I, d - II
B) a - II, b - III, c - IV, d - I
C) a - III, b - IV, c - II, d - I
D) a - IV, b - III, c - II, d - I
E) a - I, b - II, c - III, d - IV

5. En el programa areas, la sentencia para la llamada al procedimiento lectura es:

```
program areas;
uses crt;
var
    base,altura,radio:integer;
    procedure lectura(base, altura, radio:integer);
    begin
        write('Introduce datos de entrada: ');
        readln(base, altura, radio);
    end;
  function circulo(radio:integer):real;
      begin
        circulo:=pi*radio*radio;
      end;
  function perimetro(base, altura:integer):integer;
    begin
       perimetro:=2*(base+altura);
    end;
  function area(base, altura:integer):integer;
      begin
        area:=base*altura;
      end;
  begin       { inicia programa principal}
    _____________________________;
    writeln(circulo(radio));
    writeln(perimetro(base, altura));
    writeln(area(base, altura));
    readln;
  end.
```

A) **lectura;**
B) **lectura(altura);**
C) **lectura (base);**
D) **lectura(base, altura);**
E) **lectura(base, altura, radio);**

Respuesta a los reactivos muestra.

1) A, 2) B, 3) B, 4) C, 5) E.

Bibliografia.

Joyanes , Aguilar Luis, Programación en Turbo Pascal versión 5.5, 6.0 y 7.0.
Editorial McGraw-Hill.
Byron S. Gottfried. Programación en Pascal. Serie Shaum.
Editorial McGraw-Hill.
Stephen K. O´Brien, Steve Nameroff. Turbo Pascal 7.
Editorial McGraw-Hill.
Julien Hennefeld. Turbo Pascal con aplicaciones 4.0, 6.0.
Grupo Editorial Iberoamerica.
Leobardo López R. Programación Estructurada en Turbo Pascal 7.
Editorial Alfaomega.
Mat. Gilberto Fuentes Romero. Cibernética y computación II, Lenguaje de Programación Pascal. C.C.H. Plantel Sur.

Unidad IV. Estructuras de datos definidos por el usuario.

Introducción

Los problemas que resolviste en las unidades anteriores, utilizaste los tipos de datos primitivos (integer, real, char y bolean). En esta unidad, utilizaras las estructuras de datos definidos por el usuario, como una extensión de los tipos mencionados, mediante el planteamiento y solución de problemas que involucran a las estructuras de datos arreglos, cadenas, registros y archivos.

Propósito. Al final de la unidad, utilizarás las estructuras de datos mediante el desarrollo de programas, para generalizar los tipos de datos primitivos.

Aprendizajes que lograrás en la unidad.

- Describe las características de las estructuras de datos de tipo arreglo, enumerado, subrango, cadena, conjunto, registro y archivo.
- Explica la declaración de tipos de datos: arreglo, cadena, registro, archivos y la forma de acceder a los elementos de los mismos.
- Describe las funciones y procedimientos para el manejo de cadenas y archivos.
- Explica las diferencias entre archivos de acceso secuencial y directo.
- Elabora programas que involucran los tipos de datos.
- Conoce la diferencia entre estructuras estáticas y estructuras dinámicas.

Estrategias de aprendizaje.

- ✓ Investigación bibliográfica sobre las estructuras de datos definidos por el usuario: arreglos, cadenas, registros y archivos.
- ✓ Resolución de problemas que involucran las estructuras de datos definidos por el usuario.
- ✓ Construcción de programas que involucran las estructuras de datos definidos por el usuario.

Arreglos

Un arreglo es una colección de casillas de memoria para almacenar una lista de valores que debe ser del mismo tipo. La colección se identifica con un nombre y las casillas con el nombre de la colección y un índice entre corchetes. En los arreglos letras y enteros, la primera columna, representa los arreglos letras y enteros, respectivamente, la segunda columna representa los índices de cada casilla en ambos arreglos y la tercera columna, representa el acceso a algunas casillas de los arreglos mencionados. Asimismo, en el arreglo frutas, las casillas sombreadas representan el arreglo frutas, los renglones 1 y 2 representan los índices para los renglones, mientras que las columnas 1, 2 y 3 representan los índices para las columnas.

Actividad de aprendizaje. Completa el acceso a las casillas en los tres arreglos.

letras	indice	acceso a las datos
b	1	letras[1]
o	2	
p	3	letras[3]
e	4	
r	5	letras[5]
m	6	
a	7	letras[7]
s	8	
d	9	letras[9]
x	10	

enteros	indice	acceso a los datos
78	1	enteros[1]
34	2	
12	3	enteros[3]
9	4	
25	5	enteros[5]

frutas

	1	2	3
1	manzana	plátano	uva
2	melón	guayaba	sandía

Acceso a los datos

frutas[1,1]		frutas[1,3]
	frutas[2,2]	

El acceso a las casillas de los arreglos en forma interactiva, se logra con la sentencia de ciclo for, tal como se especifica en los siguientes segmentos de código:

```
For indice:= 1 to 10 do
    letras[indice];
```

```
for indece:= 1 to 5 do
    enteros[indice]
```

```
for i:=1 to 2 do
    for j:=1 to 3 do
        frutas[i,j];
```

La dimensión de un arreglo se encuentra con el producto de renglones por columnas, por lo que la dimensión del arreglo letras es 10x1.
¿Cuál es la dimensión de los arreglos enteros y frutas? _____, _____

Los arreglos de una dimensión se les llama unidimensionales, los de dos dimensiones se les llama bidimensionales y para arreglos con m reglones y n columnas, la dimensión es mxn.

Declaración de arreglos.

Los arreglos se pueden declarar en la sección de variables (var) o declarando un identificador de tipo en la sección type y declarando después una variable de ese tipo. El propósito para crear un tipo arreglo es para declarar variables con ese tipo de estructura que pueden ser utilizadas como parámetros en funciones y/o procedimientos.

Declarando tipos y variables de tipo arreglo

```
const
      lim = 10;
      tam = 5;
      ren = 2;
      col = 3;

type letras     = array [1.. lim] of char;          {Declaración de tipos de arreglos}
      enteros   = array [1.. tam] of integer;
      frutas    = array [1.. ren, 1..col] of string[10];
var
    vocales:letras;                 {Declaración de la variable vocales de tipo letras}
    numeros:enteros;                {Declaración de la variable numeros de tipo enteros}
    tropicales:frutas;              {Declaración de la variable tropicales de tipo frutas}
```

Actividades de aprendizaje. Realiza lo que se te pide:

Describe las características de un arreglo. _____

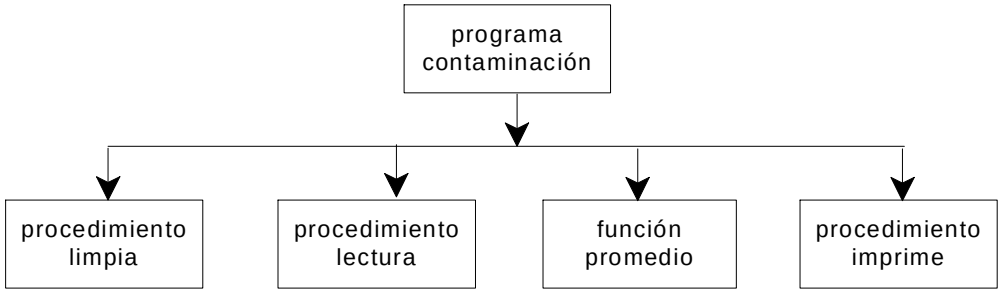
¿Cuál es la dimensión de un arreglo? _____

¿Cuál es la palabra reservada para declarar arreglos? _____

Explica cómo se accede a los elementos de un arreglo. _____

Programación con arreglos.

Problema contaminación. Encuentra el promedio de la calidad del aire para diez lecturas, toma en cuenta los procedimientos y funciones que se presentan en el siguiente diagrama:



Calidad del aire

120
116
125
115 Promedio = 122.20
121
129
127
122
119
128

Conceptos para la programación con la estructura de arreglos.

¿Cuántos módulos tiene el programa contaminación? _____

Describe la acción del procedimiento limpia. _____

Describe la acción del procedimiento lectura _____

Describe la acción de la función promedio. _____

Describe la acción del procedimiento imprime _____

¿Cuál es el propósito para declarar un tipo de estructura arreglo? _____

Describe la acción del programa principal: _____

El programa contaminacion determina el promedio de las lecturas de la calidad del aire del medio ambiente de acuerdo al diagrama presentado. Escribe en las líneas la declaración de lecturas del tipo arreglo de tamaño diez y para la variable tomas del tipo lecturas.

```
program contaminacion;
uses crt;
const
   lim = 10;
type
  _____    {aquí va la declaración de lecturas del tipo arreglo de tamaño
lim}
var
   _____  {Aquí va la declaración de la variable tomas de lecturas}
  ind:integer;
  acumula:real;

{***** PROCEDIMIENTO QUE INICIALIZA EL CONTENIDO DEL ARREGLO *****}

procedure limpia(var tomas:lecturas);
  begin
     for ind:= 1 to lim do
        tomas[ind]:=0;
```

```
{***** PROCEDIMIENTO QUE ASIGNA VALORES AL ARREGLO *****}

procedure lectura(var tomas:lecturas);
  begin
     gotoxy(20,6);
     write('LECTURA DE LECTURAS');
     for ind:=1 to lim do
       begin
          gotoxy(15,ind+7);
          write('Digita lectura ',ind,': ');
          readln(tomas[ind]);
       end;
     gotoxy(15,22);
     write('Oprime "enter" para continuar');
     readln;
  end;
```

```
{FUNCIÓN QUE CÁLCULA EL PROMEDIO DE LOS DATOS DEL ARREGLO}

function promedio (tomas:lecturas):real;
  begin
    acumula:=0;
    gotoxy(15,20);
    write('Promedio de lecturas');
    for ind:=1 to lim do
      begin
         acumula:=acumula + tomas[ind];
      end;
     promedio:=acumula/lim;
  end;
```

```
{***** PROCEDIMIENTO QUE IMPRIME LOS VALORES DEL ARREGLO *****}

procedure imprime(tomas:lecturas);
   begin
       clrscr;
       gotoxy(20,6);
       write('Contenido en LECTURAS');
       for ind:=1 to lim do
          begin
              gotoxy(15,ind+7);
              write('esribe lectura ',ind,': ');
              writeln(tomas[ind]);
          end;
       gotoxy(15,22);
       write('Oprime "enter" para continuar');
       readln;
   end;
```

```
{***** INICIA PROGRAMA PRINCIPAL *****}

begin
    clrscr;
    limpia (tomas);
    lectura(tomas);
    imprime(tomas);
    gotoxy(15,24);
    write(promedio(tomas):6:2);
    readln;
end.
```

Actividades de aprendizaje.

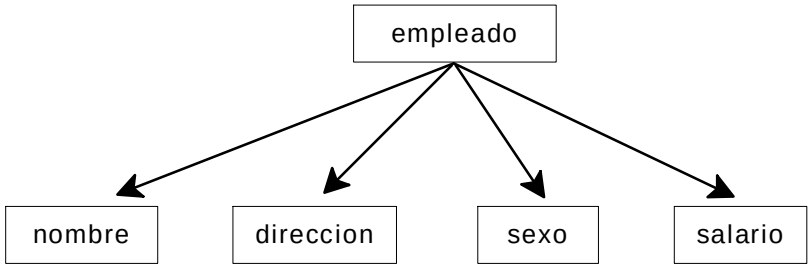
Escribe las declaraciones faltantes en el programa contaminación.

Abre el programa **ambiente.pas** que viene en el CD que acompaña a la guía, compílalo y ejecútalo.

Describe lo que hace el programa _____
_____
_____
_____

Manejado registros

Un registro es un conjunto de campos de igual o diferente tipo, se utiliza en programación para diseñar y elaborar conjuntos de registros, el siguiente diagrama ilustra cuatro campos del registro *empleado*.



El registro *empleado* tiene cuatro campos, los dos primeros son de tipo cadena con longitud diferente, el tercero es de tipo carácter y el último es de tipo numérico.

Declarando tipos de registros

Los registros se declaran en la zona de tipos de la estructura de programas en Pascal, luego se declaran variables de esos tipos, para ser utilizadas como parámetros en funciones y procedimientos del programa.

Declaración de registros

```
Type empleado = record
                    nombre   : string[25];
                    direccion : string[30];
                    sexo      : char;
                    salario   : real;
                end;

    fecha = record                                    {mes es un tipo de dato enumerado}
              mes : (enero, febrero, marzo, abril, mayo, junio, julio, agosto,
                     septiembre, octubre, noviembre, diciembre);
              dia  : 1..31;            { dia y ayo son tipos de datos subrango}
              ayo  : 2000..2010;
            end;

    articulos : record
                  descripcion      : string[15];
                  clave            : integer;
                  precio           : real;
                end;

    nomina = array[1..lim] of empleado;   {Declaración de un arreglo de registros}
```

Declaración de variables de tipo registro

```
var
   trabajador : empleado;
   fecha_nac : fecha;
   elementos : articulos;
```

Ejemplos de asignaciones a las variables de los tipos declarados

Variable trabajador:

Nombre	direccion	sexo	salario
Raquel Castro Hernández	5 de mayo, colonia centro	F	5000

Variable registro *fecha_nac*:

mes	dia	Ayo
octubre	02	1968

variable registro *elementos*:

descripcion	clave	precio
computadoras	1235	15000

El acceso a los campos de un registro, se puede hacer de dos formas, la primera, consiste en utilizar el carácter punto (**.**), como separador entre los identificadores de la variable registro y el de campo, en la segunda, se utiliza la sentencia **with**.

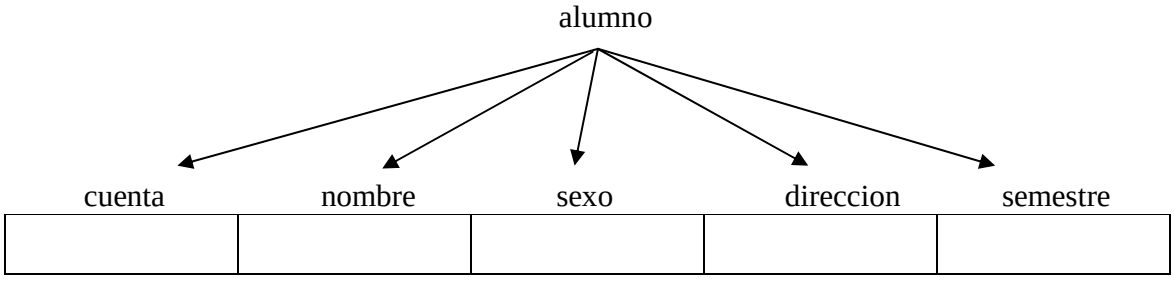
Las siguientes tablas, ilustran el acceso a algunos campos de los registros señalados en ambas formas, en la primera, se utiliza el separador (**.**), mientras que en la segunda, se utiliza la sentencia with. Para practicar el acceso a los campos de los registros mencionados, completa los accesos a los campos de los registros en ambas tablas.

trabajador**.**nombre;		
	fecha_nac**.**dia	
		elementos**.**precio
trabajador.salario;		

with trabajador do begin nombre; direccion; edad; salario; end;		

Actividad de aprendizaje

Considera el registro alumno que se presenta en el esquema y declara un tipo para el registro alumno y la variable estudiante del tipo de registro alumno.



Declaración del tipo registro alumno	Declaración de la variable estudiante del tipo registro
_____ _____ _____ _____ _____ _____ _____	_____

El programa registros utiliza alumno como un tipo registro, alumnos como un tipo arreglo unidimensional con tamaño 3 (tam=3) del registro alumno y la variable grupo del tipo alumnos. Declara los tipos y la variable mencionados

```
program registros;
uses crt;
const
   tam=3;
type _______ = _________
              cuenta   :string[25];
              nombre   :string[25];
              direccion:string[40];
              sexo     :char;
              semestre:integer
            end;
    ___________= ___________________;
var
  i:integer;
  _________________;
```

```
procedure captura(var grupo:alumnos);
  begin
     for i:=1 to tam do
      begin
         clrscr;
         gotoxy(12,6);
         write('Digita cuenta: ');
         readln(grupo[i].cuenta);
         gotoxy(12,8);
         write('Digita nombre: ');
         readln(grupo[i].nombre);
         gotoxy(12,10);
         write('Dgita dirección: ');
         readln(grupo[i].direccion);
         gotoxy(12,12);
         write('Digita sexo : ');
         readln(grupo[i].sexo);
         gotoxy(12,14);
         write('Digita semestre : ');
         readln(grupo[i].semestre);
       end;
  end;
```

```
procedure imprime(grupo:alumnos);
  begin
     for i:=1 to tam do
      begin
         clrscr;
         gotoxy(12,6);
         write('Cuenta: ');
         write(grupo[i].cuenta);
         gotoxy(12,8);
         write('Nombre: ');
         writeln(grupo[i].nombre);
         gotoxy(12,10);
         write('Dirección: ');
         writeln(grupo[i].direccion);
         gotoxy(12,12);
         write('Sexo : ');
         writeln(grupo[i].sexo);
         gotoxy(12,14);
         write('Semestre: ');
         writeln(grupo[i].semestre);
         gotoxy(12,16);
         write('Digita enter para Continuar');
         readln;
       end;
  end;
```

```
Begin                {****  Inicia el programa principal  ****}
   captura(grupo);
   imprime(grupo);
end.
```

Actividades de aprendizaje.

Al finalizar la declaración de alumno, alumnos y grupo, procede a la edición, compilación y ejecución del programa registros y realiza lo que se te indica:

Explica la acción que realiza el procedimiento captura _____
_____
_____

Explica la acción que realiza el procedimiento imprime _____
_____
_____

Explica la acción que realiza el programa principal _____
_____
_____

El programa registros utiliza el punto (**.**), para acceder a los campos del registro. Recuerda que también se puede utilizar la sentencia **with**. Utiliza la sentencia **with grupo[i] do**, después de begin de la sentencia for en los procedimientos captura e imprime. En el procedimiento captura, borra grupo[i] y el punto que sigue, para el procedimiento imprime, borra, grupo[i] y el punto que sigue.

Actividad de aprendizaje.

Realiza la edición del programa registros con las modificaciones pertinentes, para que considere la sentencia with, después compílalo y ejecútalo.

Conceptos sobre la estructura registros

Describe las características de un registro _____
_____

Escribe la palabra reservada para declarar registros _____

Explica cómo se accede a los campos de un registro _____
_____

Menciona la finalidad de declarar tipos de registros _____
_____

Manejando archivos

Los tipos de datos de entrada y salida, utilizados en los programas anteriores, podemos decir que los de *entrada*, fueron introducidos de manera interactiva, es decir, desde el teclado, quedando almacenados de manera temporal en la memoria principal de la computadora, de ésta, fueron tomados por la unidad aritmético - lógica, bajo la supervisión de la unidad de control, para producir los datos de *salida*, que fueron visualizados en pantalla y almacenados de forma temporal en la memoria principal de la computadora, perdiéndose ambos al finalizar la ejecución del programa, sin embargo, en programación, es común el almacenamiento de ellos en dispositivos de almacenamiento secundario, tales como discos flexibles y/o duros, para que aplicaciones posteriores puedan utilizarlos como datos de entrada, desde estos dispositivos de almacenamiento secundario, simplificándose la tarea de introducir datos en forma interactiva.

¿Qué es un *archivo*? _____
_____

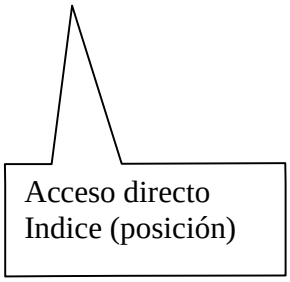
¿Cuáles tipos de *archivos* conoces? _____
_____

Escribe cinco ejemplos de *archivos* de aplicaciones diferentes: _____
_____

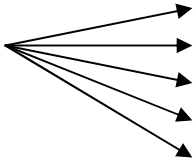
Los archivos que trataremos en la presente unidad, son los llamados *tipificados (archivos binarios)*, y los de tipo *texto*, los *primeros*, almacenan cualquier tipo de datos simple y/o estructurados (arreglos, registros, etcétera), excepto archivos, los primeros, sólo pueden ser escritos y leídos con programas en Pascal y el acceso a ellos, puede ser secuencial o aleatorio, mientras que los *segundos*, almacenan caracteres, cadenas de caracteres, enteros, reales, los cuales pueden ser escritos y leídos con algún editor de textos o con un programa elaborado con algún lenguaje de programación y el acceso a ellos es de manera secuencial. El acceso aleatorio a cualquier registro en archivos tipificados, se logra de forma directa, sin tener que pasar por los registros anteriores a éste, tomando en cuenta la posición que guarda en el archivo, mientras que el acceso a los archivos de texto es de manera secuencial, es decir, la lectura se hace registro por registro. Cabe señalar que Pascal, enumera a los registros de los archivos, siendo 0 el primero de ellos. El siguiente esquema, presenta en forma gráfica el acceso a los registros en los archivos tipificados y de texto, así como la enumeración de los mismos.

Archivo tipificado: *empleado*

Registro	Nombre	Dirección	Edad	Salario
0				
1				
2				
3				
4				
5				



Acceso secuencial



Archivo texto: vocales

Registro	**Letras**
0	a
1	e
2	i
3	o
4	u

Declarando tipos de archivos tipificados.

Un archivo tipificado lo podemos ver como un conjunto de registros, un registro como un conjunto de campos, agrupados en un identificador de tipo registro, declarado en la zona para la declaración de tipos.

Declaración de archivos tipificados y variables de los mismos

```
const lim = 10;

type lecturas = array [1..lim] of integer;

     empleado = record
                        nombre   : string[25];
                        direccion : string[30];
                        sexo      : integer;
                        salario    : real;
                   end;

     articulos : record
                       descripcion : string[30];
                       clave         : integer;
                       precio        : real;
                 end;

      archlect = file of lecturas;
      archempl : file of empleado;          {Declaración de tipos de archivos}
      archart : file of articulos;
```

Declaraciones de variables de tipo archivo

```
var
   trabajador : archempl;
   tabla      : archlect;          {Declaración de variables de archivo}
   elementos : archart;
```

Actividad de aprendizaje. Abre el programa **pagonet2.pas** que se encuentra en el directorio programas del cd anexo, compílalo, ejecútalo y describe la acción de los procedimientos:

Procedure menu _____
_____

Procedure archivos _____
_____

Procedure altas _____
_____

Procedure cambios _____
_____

Procedure bajas _____
_____

Procedure consultas _____
_____

Procedure archivos _____
_____

Describe el programa principal _____
_____

Actividad de aprendizaje. Consulta en un libro de programación Pascal y/o ayuda en línea del compilador de Pascal y describe los procedimientos y funciones, para la manipulación de archivos. Los primeros cinco son procedimientos y las restantes funciones.

Assign: _____
_____

Rewrite: _____
_____

Reset: _____
_____

Seek: _____
_____

Close: _____
_____

Eof: _____
_____

Filesize: _____
_____

Filepos: _____
_____

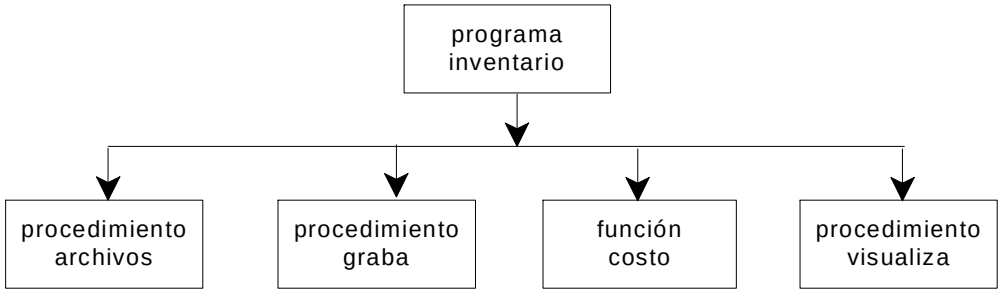
while not eof(archivo) _____
_____

{$i-} _____

{$i+} _____

Ioresult _____
_____

Actividad de aprendizaje. Elabora un programa para crear un archivo inventario con los textos de una librería, así como calcular e imprimir el inventario y valor total del mismo. Los campos del registro libro: título, autor, número de código, precio y cantidad.



Respuestas a los conceptos sobre registros

La palabra reservada para declarar registros es Record.

El acceso a los campos de un registro se obtiene de dos formas, una de ellas consiste en separar con un punto el identificador de registro y el campo, la otra consiste en especificar la palabra reservada with seguida del identificador de registro, la palabra reservada do y la lista de campos del mismo.

Acceso a los elementos de los registros empleado y alumno.

empleado.numero, empleado.nombre, empleado.sexo, etc.

alumno.no_cta, alumno.nombre, alumno.sexo, etc.

Acceso a los elementos de los registros empleado y alumno con with.

```
With empleado do          with alumno do
    Numero                    no_cta
    Nombre                    nombre
    Sexo                      sexo
    Estado_civil              direccion
    salario                   semestre
```

La finalidad de declarar tipos de registros es con la idea de declarar variables de ese tipo de registros para su manejo.

Respuestas a los conceptos sobre archivos

Los tipos de archivos que se manejan en Pascal son de texto, tipificados y no tipificados.

Los archivos de tipo texto (text) son secuenciales, mientras que los segundos son aleatorios (file of), el acceso a los primeros es secuencial y el acceso a los segundos es aleatorio.

La palabra reservada para declarar archivos de tipo texto es file of text, mientras que para los archivos de tipo tipificado es file of <tipo>.

El acceso a los elementos de un archivo se logra con la sentencia de lectura o escritura del archivo y el registro separándolo con un punto del campo o bien con la palabra reservada with y el nombre del campo.

La finalidad al declarar tipos de archivos es definir variables con ese tipo de archivos y facilitar su manejo.

El procedimiento assign tiene la operación de asignar un archivo para establecer una correspondencia entre una variable de tipo archivo (interno) con un archivo externo situado en un disco.

El procedimiento rewrite crea y abre un nuevo archivo. Si el archivo ya existe, rewrite borra su contenido, en caso contrario el archivo queda abierto para una operación de escritura.

El procedimiento close tiene como función la operación de cerrar un archivo.

El procedimiento reset es un procedimiento que tiene como función abrir un archivo existente para una operación de lectura.

La función eof es una función lógica que devuelve el valor de verdadero cuando se ha alcanzado la marca de fin del archivo, en caso contrario devuelve falso.

La sentencia while not eof(archivo) do, significa mientras no sea fin de archivo, hacer.

Ejemplos de reactivos para el examen.

1. Los siguientes enunciados son correctos, excepto.

A) Los arreglos son estructuras de datos homogéneos.

B) Las cadenas almacenan datos volátiles.

C) Los registros son los elementos básicos para la creación de archivos tipificados.

D) Los archivos son estructuras con datos dinámicos.

E) Todos los archivos en Pascal son de acceso directo.

2. La palabra reservada para declarar un registro es

A) Array.
B) String.
C) Record.
D) File.
E) Set.

3. Relaciona las columnas procedimientos y descripción.

Procedimientos	descripción
a. Assign	I Procedimiento para preparar a un archivo para escritura.
b. Reset	II Procedimiento para cerrar un archivo.
c. Rewrite	III Procedimiento para establecer una relación entre el archivo interno y externo.
d. Close	IV Procedimiento para preparar un archivo para lectura.

A) a - III, b - IV, c - II, d - I
B) a - II, b - IV, c - I, d - III
C) a - I, b - II, c - III, d - IV
D) a - III, b - IV c - I d - II
E) a - IV, b - III, c - I, d - II

4. En el siguiente segmento de programa. Escribe las declaraciones para los tipos registro amigo y el archivo amigos con la estructura de amigo; las variables persona de tipo amigo y archivo de tipo amigos.

```
program conocidos;
uses crt;
type _________ =  record
                      nombre:string[25];
                      direccion:string[40];
                      edad: string[2];
                      salario:real;
                   end;
        amigos  = _____________;
var
    sigue:char;
    persona : _________;
    _________ : amigos;
```

A) amigo, file of amigo, amigo, archivo
B) record, amigo, persona, amigos
C) type, record, file, amigo,
D) archivo, amigo, amigos, file of amigo
E) type, file of amigo, persona, amigos

5. Escribe las sentencias faltantes en el procedimiento graba.

```
procedure graba (var archivo:amigos; _________:amigo);
  begin
     __________ (archivo,'amigo.dat');
     __________(archivo);
     sigue:='s';
     while (sigue = 's') or (sigue = 'S') do
       begin
          write('Digita nombre: ');
          readln(persona.nombre);
          write('Digita dirección : ');
          readln(persona.direccion);
          write('Digita edad : ');
          readln(persona.edad);
          write('Digita salario : ');
          readln(persona.salario);
          write(archivo,persona);
          Write('Mas amigos s/n: ');
          readln(sigue);
       end;
     close(__________);
  end;    {fin procedimiento reporte}
```

A) Archivo, reset, assign, persona
B) persona, assign, reset, archivo
C) assign, persona, reset, archivo
D) archivo, persona, assign, reset,
E) reset, persona, assign, archivo

6. En el procedimiento imprime, ¿cuál es la secuencia de sentencias para visualizar en pantalla los registros del archivo amigo.dat?

```
procedure imprime (var archivo:amigos;persona:amigo);
  begin
     clrscr;
     __________ (archivo,'amigo.dat');
     __________ (archivo);
     while not eof(archivo) do
       begin
          _____________ (archivo,persona);
          _____________ ('Registros del archivo');
          _____________ ('Nombre    = ',persona.nombre);
          _____________ ('Dirección = ',persona.direccion);
          _____________ ('Edad      = ',persona.edad);
          _____________ ('Salario   = ',persona.salario:8:2);
          delay(2000);       { Retardo en milesegundos}
       end;
     close(archivo);
  end;    {fin procedimiento reporte}

begin
   graba(archivo,persona);
   imprime(archivo,persona);
end.
```

A) reset, assign, read, rewrite
B) assign, reset, read, writeln
C) assign, reset, rewrite, read
D) writeln, assign, reset, read
E) reset, assign, read, writeln

Respuestas:

Pregunta	1	2	3	4	5	6
Respuesta	E	C	D	A	B	B

BIBLIOGRAFÍA

- JOYANES, Aguilar Luis. *Programación en Turbo Pascal/Borland Pascal 7*, Madrid, Mc-Graw-Hill, 1998.
- LÓPEZ, R. Leobardo.- *Programación estructurada*. *Turbo Pascal 7*. México, Computec, 1993.
- PASCUAL, González Francisco. *Domine Turbo Pascal 6*, Madrid, RA-MA, 1992.
- SCHNEIDER, Michael G. et al. *Introducción a la Programación y solución de Problemas con Pascal*, México, Limusa Noriega, 1990.
- FUENTES, Romero Gilberto. Propuesta educativa para la materia de Cibernética y Computación II, México, CCH, Plantel Sur, 2000.

Cibernética y Computación II, Unidad V: Introducción a la programación con Delphi

Introducción

En esta unidad obtendrás una visión de la programación orientada a objetos, como una extensión de los lenguajes de programación estructurada. Asimismo, conocerás los elementos de la programación orientada a objetos del lenguaje Delphi y las diferencias entre los elementos del lenguaje de programación Pascal.

Propósito de la unidad:

Al finalizar la unidad, conocerás el manejo del lenguaje Delphi, para ampliar la visión de los lenguajes, mediante la exploración y presentación de programas.

Estrategias de enseñanza:

Investigación sobre los lenguajes de programación visuales
Planteamiento y solución de problemas.
Exploración de los componentes básicos del lenguaje Delphi.

Aprendizajes:

Describe las características de la programación orientada a objetos
Describe los elementos básicos del lenguaje Delphi.
Describe los componentes básicos del lenguaje Delphi.
Desarrolla proyectos con el lenguaje Dephi.

Evaluación:

Descripción de las características de la programación orientada a objetos.
Descripción de los componentes básicos del lenguaje Delphi.
Resolución de los problemas planteados

Recursos didácticos:

Computadoras personales.
CD del compilador de Delphi.
CD que contiene ejemplos de problema s resueltos.
Tutorial en línea del lenguaje Delphi.

Actividades de aprendizaje:

Describe las características de la programación orientada a objetos: _____
_____
_____
_____

Describe las características del lenguaje de programación Delphi: _____
_____
_____

Describe los siguientes conceptos:

Clase: _____
_____

Herencia: _____
_____

Polimorfismo: _____
_____

Objeto: _____
_____

Método: _____
_____

Instancia: _____
_____

Evento: _____
_____

Explorando los componentes de Delphi .

- Ejecuta el entorno integrado de desarrollo (IDE) de Delphi.

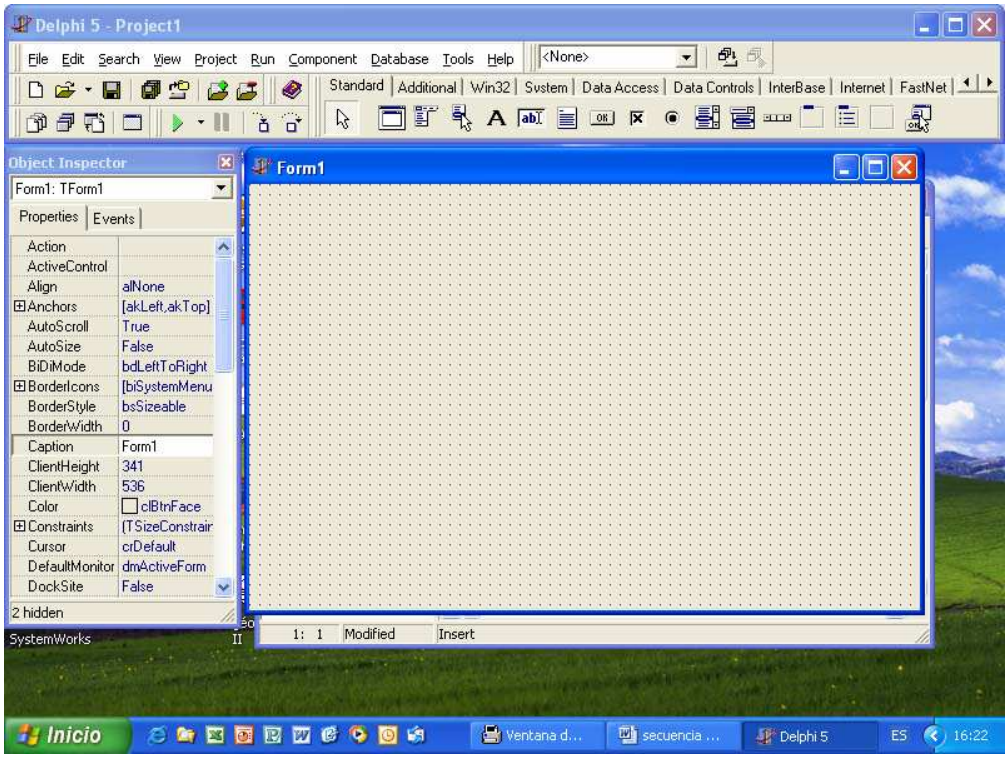


Figura 1. Entorno integradio de desarrollo.

Pasa el apuntador del mouse sobre las imágenes de los componentes Standard para que te des una idea de su descripción, tantas veces como sea necesario.

¿Cuáles son elementos del entorno integrado de desarrollo de Delphi? _____
_____

Describe los siguientes elementos de Delphi:

Formulario (Form1): _____
_____

Inspector de objetos (Object inspector): _____
_____

Propiedades (properties): _____
_____

Eventos (Events): _____
_____

Explorando los controles de la paleta de componentes estándar

Arrastra al formulario los componentes de la figura 2 y descríbelos:

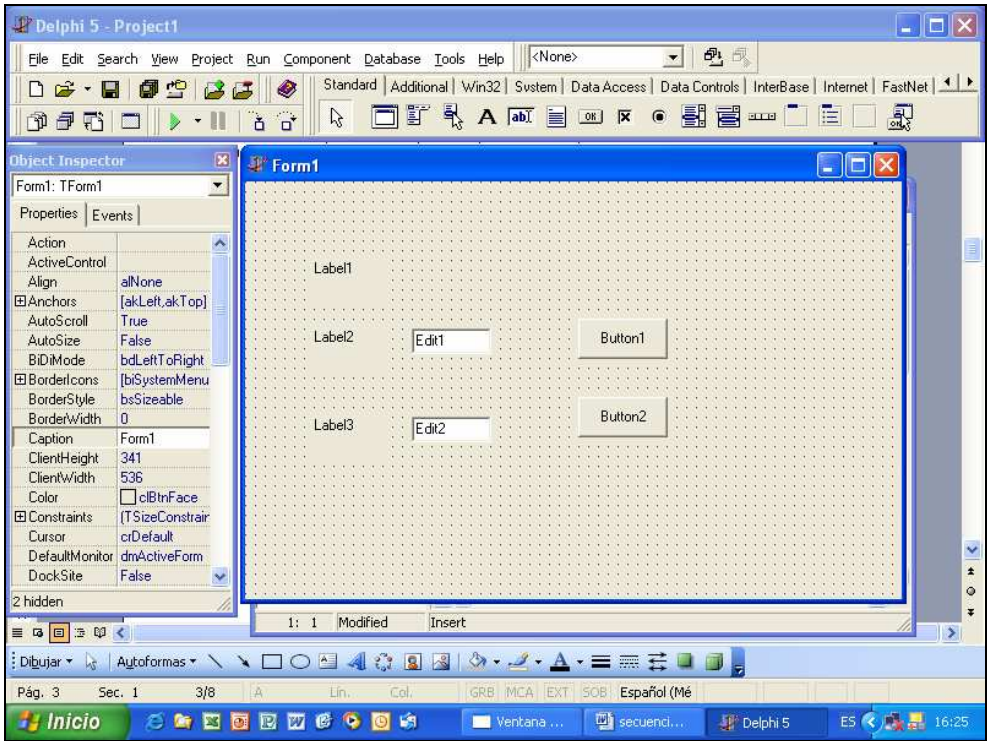


Figura 2. Controles estándar de la paleta de componentes.

Label1: _____
_____

Edit11: _____
_____

Button1: _____
_____

Escribe en la tabla los controles que pertenecen a los objetos que se especifican.

TLabel	TEdit	TButton

Explorando el inspector de propiedades

Cambia el nombre a los controles de TLabel y a los de TEdit, por los que aparecen en la tabla y para los controles TButton, selecciona los eventos que especifican en la misma.

TLabel	TEdit	TButon	Evento
etiqueta1	numero	Calcular	OnClick
etiqueta2	suma	Limpiar	OnClick
etiqueta3			

Para ello, da clic en la instancia etiqueta1, ¿qué observas en el inspector de propiedades (Caption)? _____, cambiála por la etiqueta *Problema: La suma de los n números naturales*. De la misma forma, da clic en la instancia Edit1, ¿qué obvervas en el inspector de propiedades (Text)? _____, cambiálo por espacios.

Aplica el diseñador del formulario, para que las intancias restantes de TLabel, TEdit y TButton, corresponan a las presentadas en la figura 3.

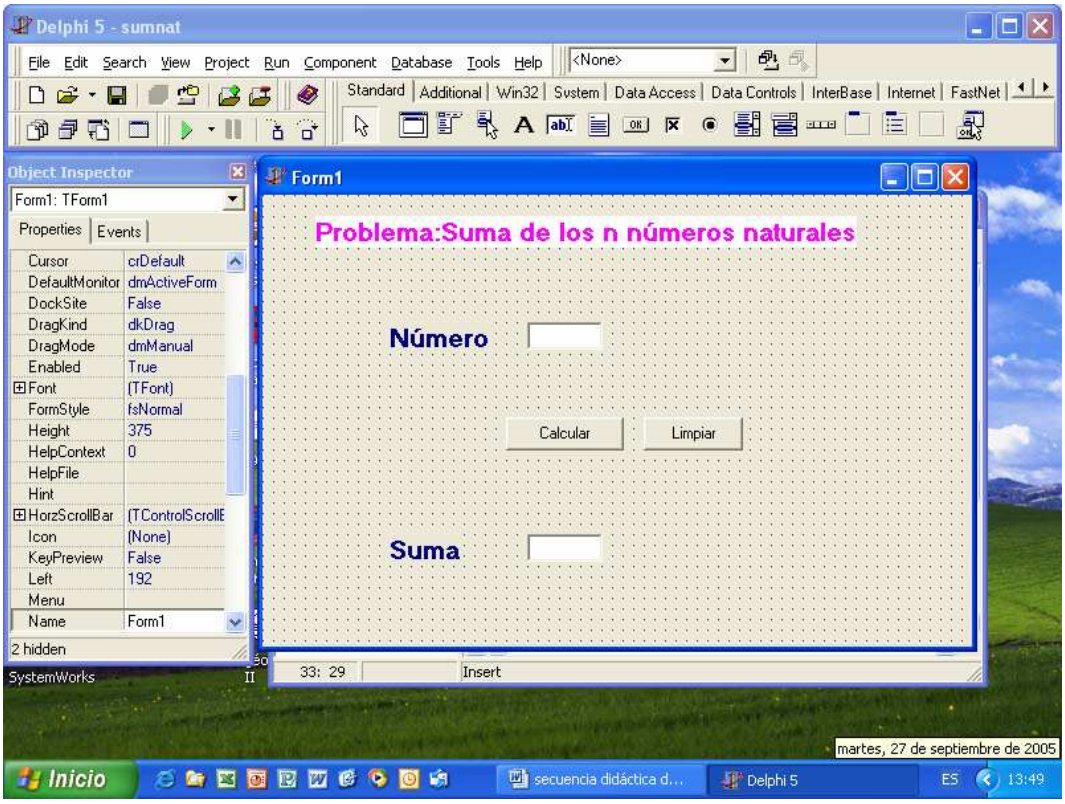


Figura 3. Diseño del formulario al aplicar propiedades a las instancias de TLabel, TEdit y TButton.

Recuerda que para cambiar las propiedades y eventos para cualquier control, debes hacer clic en la instancia del control que se encuentra en el formulario y en la columna propiedades (propierties), puedes cambiar las propiedades permitidas para el control (Caption, Name, Font, Color, etc), mientras que en la columna de eventos (Events), puedes elegir la acción que realizará el control. Asigna otras propiedades a los controles del formulario, al final, no se te olvide dejarlos como los de la figura 3.

Ejecuta el programa (F9 o con el botón ), digita en la ventana de ejecución de la aplicación el número 6. Da clic en el botón Calcular y luego en el botón Limpiar.

¿Qué resultado se obtiene? _____

¿Por qué? _____
_____

¿Qué se necesita para que el programa calcule la suma de los primeros 6 números naturales? _____

Da doble clic en el botón Calcular y desplázate hacia arriba mediante las barra de desplazamiento hasta unit Unit1.

Describe los que observas: _____
_____
_____

Observación.

Es pertinente mencionar que lo realizado hasta el momento lo ha desarrollado Delphi, mediante el arrastre que hiciste de los controles al formulario y la aplicación de propiedades a los mismos. Por lo que falta la acción del evento OnClick para lanzar el resultado. Está acción se traduce en el método (código) para determinar la suma de los n números naturales. De seguro, este código, ya lo trabajaste en alguna unidad anterior y se presenta en la figura 4, al igual que el código del evento del procedimiento TForm1.LimpiarClick, edítalos tal como se aprecia en la figura mencionada.

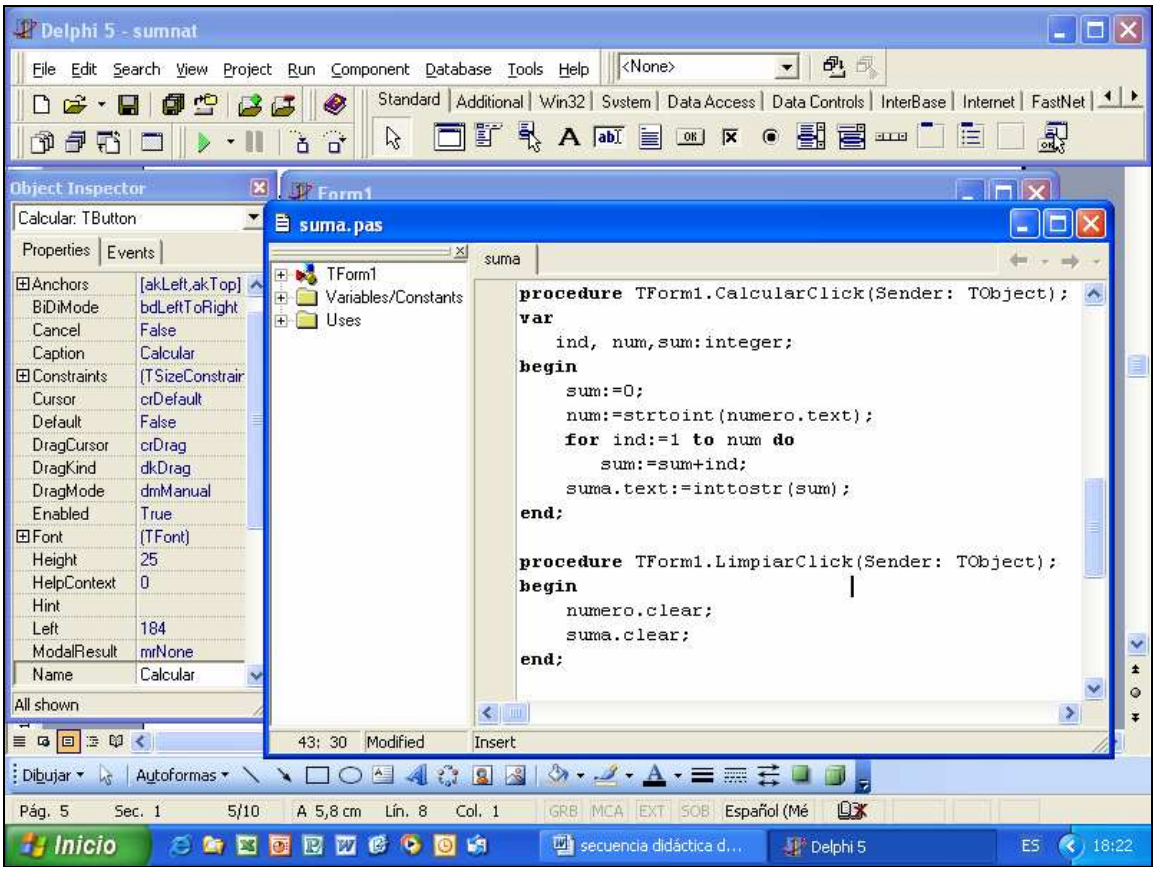


Figura 4. Código para el evento OnClick

Ejecuta el programa, digita en la ventana de edición el número 6. Da clic en el botón Calcular y luego en el botón Limpiar.

¿Qué resultado se obtiene? _____

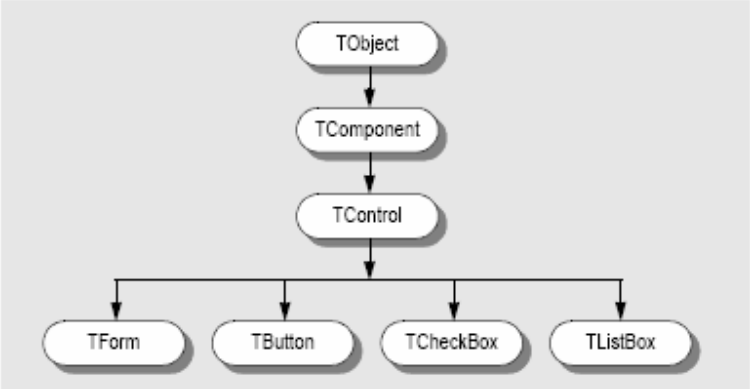
¿Es correcto? _____. ¿Por qué? _____
(Si, No)

Observaciones:

Strtoint Convierte un texto a valor numérico. Es necesario para operar con él, ya que los TEdit reciben valor alfanumérico. Podría haber utilizado val.

Inttostr Convierte un valor numérico a texto. Es necesario para operar con él, ya que los TEedit reciben valor alfanumérico. Podría haber utilizado str.

El desarrollo de proyectos con Delphi está relacionado con el manejo de objetos, componentes, controles, formas, botones, etiquetas, cuadros de texto y cuadros de imágenes, entre otros, tal como se presenta en el siguiente esquema:



Actividades de retroalimentación

- Realiza las adecuaciones al problema suma de naturales, mediante la sentencia while.
- Realiza las adecuaciones pertinentes al problema suma de naturales, mediante la sentencia repeat.
- Desarrolla el proyecto calculadora que se presenta en la figura5.

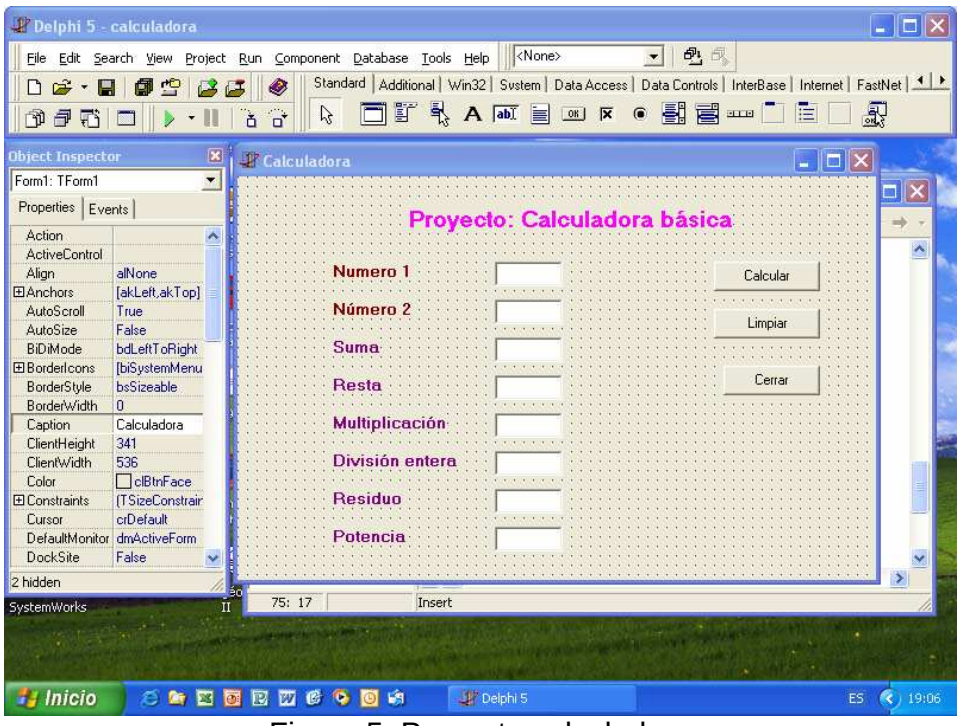


Figura 5. Proyecto calculadora

Identifica los controles del formulario de la figura 4 y escríbelos en la tabla siguiente:

TLabel	TEdit	TButton	Events

Escribe el código para los controles de botón Calcular, Limpiar y Cerrar. Al ejecutar programa con los datos para los números 15 y 4, se obtiene los resultados que aparecen en la ventana de ejecución.

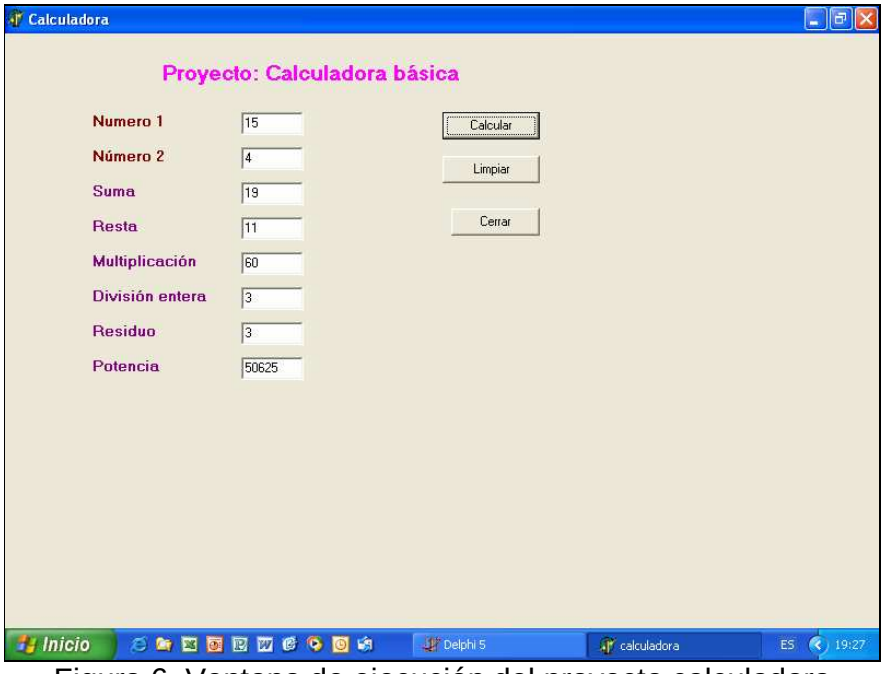


Figura 6. Ventana de ejecución del proyecto calculadora

Reactivos muestra

1. **Relaciona de forma correcta ambas columnas.**

a. Objeto I. Es una aplicación.
b. Evento II. Es una forma o un cuadro de diálogo.
c. Formulario III. Es la respuesta a un objeto.
d. Proyecto IV. Está compuesto por datos y métodos que los manipulan.

A) a - III, b - IV, c - II, d - I
B) a - I, b - III, c - IV, d - II
C) a - IV, b - III, c - II, d - I
D) a - II, b - I, c - III, d - IV
E) a - IV, b - II, c - I, d - III

2. **Relaciona las columnas entre objetos y descripción de forma correcta.**

a. Permite el diseño de la interfaz gráfica para la aplicación.	I. Paleta de componentes
b. Conjunto de controles para se utilizados es las aplicaciones	II. Inspector de objetos
c. Permite cambiar las características de los objetos contenidos en el formulario.	III. Editor de código
d. Permite realizar la acción de los eventos para los controles del formulario.	IV. Formulario

A) a - III, b - IV, c - II, d - I
B) a - IV, b - I, c - II, d - III
C) a - I, b - III, c - IV, d - II
D) a - II, b - I, c - III, d - IV
E) a - IV, b - II, c - I, d - III

3. **Asocia de forma correcta ambas columnas.**

a. Clase I. Significa que los objetos diferentes derivados del mismo ancestro soporta el mismo método y propiedades de interface.
b. Instancia II. Construcción de una jerarquía de objetos descendientes, donde cada objeto descendiente hereda de todos sus ancestros y, tiene acceso a los datos y métodos de sus descendientes.
c. Herencia III. Está integrado por datos y los métodos que lo manipulan.
d. Polimorfismo IV. Es una referencia del objeto.

A) a - III, b - IV, c - II, d - I
B) a - I, b - II, c - III, d - IV
C) a - II, b - IV, c - I, d - III
D) a - IV, b - I, c - IV, d - II
E) a - III, b - IV, c - I, d - II

4. Asocia de forma correcta el componente y su descripción.

a. Cuadro de texto	I. Objeto que permite iniciar acciones.
b. Etiqueta	II. Objeto para desplegar una lista enrollada de selección.
c. Caja de lista	III. Objeto para desplegar texto que no puede ser seleccionado o manipulado.
d. Botón	IV. Objeto que permite introducir y modificar texto.

A) a - I, b - III, c - IV, d - II
B) a - I, b - II, c - III, d - IV
C) a - II, b - IV, c - I, d - III
D) a - IV, b - III, c - II, d - I
E) a - III, b - II, c - I, d - IV

5. Selecciona las sentencias del procedure TForm1.calcularClick(Sender: TObject), para determinar los cálculos especificados de dos números enteros.

```
procedure TForm1.calcularClick(Sender: TObject);
_________
_________
begin
   ____________________________________
   ____________________________________
   suma.text:=inttostr(num1+num2);
   resta.text:=inttostr(num1-num2);
   multiplicacion.text:=inttostr(num1*num2);
   division.text:=inttostr(num1 div num2);
   residuo.text:=inttostr(num1 mod num2);
   pot:=round(exp(ln(num1)*num2));
   potencia.text:=inttostr(pot);
end;
```

er;

num2:=strtoint(numero2.text);
var
num1, num2 , pot: integer;

var
num1:=strtoint(numero1.text);
num2:=strtoint(numero2.text);

C)
```
var
num1, num2 , pot: integer;
num1:=strtoint(numero1.text);
num2:=strtoint(numero2.text);
```

D)
```
num1:=strtoint(numero1.text);
num2:=strtoint(numero2.text);
num1, num2 , pot: integer;
var
```

E)
```
var
num1:=strtoint(numero1.text);
num1, num2 , pot: integer;
num2:=strtoint(numero2.text);
```

6. Selecciona las sentencias del procedure TForm1.botonClick(Sender: TObject), para calcular la suma de los n primeros números naturales.

```
procedure TForm1.botonClick(Sender: TObject);
var
n:word;
indice,sum:integer;
begin
    n:= strtoint(num.text);
    indice:=1;
    sum:=0;
    _____________________________
    Suma.caption:='Suma = '+inttostr(sum);
end;
```

A)
```
begin
    sum:=sum+indice;
    indice:=indice+1;
end;
```

```
begin
    sum:=sum+indice;
    indice:= indice +1;
end;
```

C)
```
repeat
    sum:=sum+indice;
    indice:=indice+1;
until (indice <= n);
```

B)
```
while (indice > n do
    begin
        sum := sum  + indice;
    indice:=indice;
end;
```

E)
```
if  indice <= n ) then
    begin
        sum:=sum+indice;
        indice:=indice+1;
    end;
```

Resuesta a reactivos muestra

Reactivo	1	2	3	4	5	6
Respuesta	C	B	A	D	C	A

BIBLIOGRAFÍA

- CANTÚ, Marco. *Delphi (Kylix) 7*, Madrid, Anaya Multimedia, 2002.
- CHARTE, Ojeda Francisco, *Guía Práctica para Usuarios de Delphi 7*, Madrid, Anaya Multimedia, 2002.
- SENS, Herbes. *El gran libro de Delphi*, Marcobo, 1996.

BIBLIOGRAFÍA ELECTRONICA (Las paginas de Internet no son permanentes)

- http://www.marcocantu.com. En esta página encontraras un tutorial para Delphi.

AUTORES

Unidad 1: Lenguaje de Programación Pascal
Helios Becerril Montes

Unidad 2: Estructura de Control de Secuencia
Christyan Mabel Mendoza Martínez y Rosalba Rodríguez Maldonado

Unidad 3: Procedimientos y Funciones
José Mateos Cortés y Jesús Ramírez Vega

Unidad 4: Estructuras de Datos Definidos por el Usuario
Gilberto Fuentes Romero

Unidad 5: Introducción a la Programación en Delphi (Kylix)
Gilberto Fuentes Romero

Coordinación:
Gilberto Fuentes Romero y Asunción Reynoso Díaz