
	<p>UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

PROYECTO VALOR 4%

DEFENSA VALOR 3%

Proyecto Pascal.

OBJETIVO

Poner en práctica los conocimientos adquiridos durante el curso, en particular algunas de las ideas de programación que derivan de los análisis léxicos, sintácticos y semánticos de los lexemas que conforman las hileras de caracteres que se usan, como código fuente, como entrada del programa.

DESARROLLO

PAZCAL es una simulación básica, ficticia, del lenguaje de programación PASCAL (versión estándar o clásica) y como tal es un poco más rudimentario, careciendo de muchas de las características de su hermano mayor.

PAUTAS

- El proyecto es individual.
- Debe ser hecho en Java. No puede usarse otro lenguaje.
- El entorno de desarrollo debe ser Netbeans, oficial de la universidad. No se aceptará entregables en otro IDE. Se debe de utilizar mínimo la versión 8, se sugiere utilizar la última versión.
- Debe ser programado en modo carácter.
- No se debe utilizar librerías de terceros para los instrumentos de evaluación solicitados en este curso. Una librería de terceros son fragmentos de códigos que otras personas han desarrollado y lo encapsulan en un componente, el cual podría ser utilizado como una referencia en la solución del problema vía código fuente
- Se recomienda primero que se retroalimente del programa PASCAL, para lo cual es importante que lean los siguientes enlaces:



1. Historia del lenguaje PASCAL y ubicación del contexto histórico

<https://puntoequis.com.ar/donde-esta-pascal-hoy/>

<https://lenguajesdeprogramacion.net/pascal-y-delphi/>

2. Este es el enlace donde se puede descargar el FREE PASCAL (seleccionar su plataforma y eso los lleva a la página de descargas):

<https://www.freepascal.org/download.html>

	<p>UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

3. Usaremos como “ejecutador” el programa FREE PASCAL; aquí tenemos la guía del usuario donde se explica cómo compilar desde la línea del CMD (capítulo 3: **Compilador Usage**):

<https://www.freepascal.org/docs-html/current/user/user.html>

4. Otra documentación interesante y apropiada se puede ver a la derecha donde dice "Documentation" según se ve en este enlace:

<https://www.freepascal.org/docs.html>

Adicionalmente, se les recuerda que este cuatrimestre no habrá libro de texto, sino que deberán usar los recursos didácticos que se indiquen en la plataforma Moodle (en la “**Documentación General de la Asignatura**”, entre otros); además, deberán revisar semana a semana el **Foro de Consultas**, los temas de estudio, las lecturas sugeridas, los materiales anexados, etc. y estar pendientes de las **sesiones virtuales** para atender dudas, las cuales se comunicarán oportunamente en la plataforma Moodle en el foro de los “**Anuncios de Interés**”. También tienen la libertad de tomar la iniciativa y buscar en el Internet material afín a los temas del curso.

NOCIONES BÁSICAS

PAZCAL es un compilador sencillo (no genera ejecutable nativo) que permite compilar programas básicos hechos en un lenguaje similar al de PASCAL llamado PAZCAL.

Usando Netbeans (la herramienta oficial) deben realizar un programa en Java llamado PAZCAL.java que implemente un compilador para PAZCAL. Una vez programado y probado se debe generar el archivo .jar o sea el PAZCAL.jar

La sintaxis para usar el compilador PAZCAL es la siguiente, la cual se digita en la línea de comandos de MICROSOFT WINDOWS o CMD; este punto es importante porque el programa tuyo el Profesor no lo va a probar desde dentro de Netbeans sino desde aquí:

```
C:\> java -jar PAZCAL.jar ArchivoPAZCAL.PAZCAL
```

En donde ArchivoPAZCAL.PAZCAL es el nombre de archivo de texto con extensión .PAZCAL. Dicho nombre no es sensible a mayúsculas/minúsculas y debe seguir las siguientes convenciones de nombres de archivos:

- Empezar con una letra.
- Tener solo letras o números.
- No usar caracteres especiales ni siquiera el guion bajo.
- La extensión debe ser “PAZCAL” como ya se indicó.
- Si no se indica la extensión se puede asumir.

SOBRE LA INDEPENDENCIA FÍSICA DEL COMPILADOR

Es necesario insistir mucho en este punto: se debe entregar el archivo *.jar listo para ser ejecutado en cualquier máquina y en cualquier carpeta sin depender de todo el ambiente de desarrollo de Netbeans. En aquella carpeta que el Profesor disponga, por ejemplo: **D:\REVISION**, debe bastar con echar el archivo *.jar, los archivos de entrada de la revisión y tu compilador debe generar los archivos

de salida ahí mismo; se recalca una vez más: ahí mismo, no en la raíz de C:\, ni en la raíz de D:\ ni en el escritorio de MICROSOFT WINDOWS, sino en la carpeta que el Profesor disponga. En otras palabras, se desea eliminar la dependencia física de tu compilador hacia tu máquina y que se pueda ejecutar en cualquier parte.

También es importante recordar que en cada entrega deberán adjuntar la carpeta completa del proyecto realizado en netbeans.

EJEMPLO DE USO DEL COMPILADOR

Así, si tenemos el siguiente archivo de texto que realiza cálculos simples sobre la inflación, llamado:

INFLACION.PAZCAL

y cuyo contenido es el siguiente (ver próxima página):

```
PROGRAM INFLACION ( INPUT, OUTPUT );
{
  Asumiendo tasas de inflación anual de 7%, 8%, y 10%,
  encontrar el factor por el cual cualquier moneda, tales como
  el franco, el dólar, la libra esterlina, el marco, el rublo, el yen
  o el florín han sido devaluadas en 1, 2, ....., N anos.
}

VAR
  MAXANOS: INTEGER;
  ANO:     INTEGER;
  FACTOR1: REAL;
  FACTOR2: REAL;
  FACTOR3: REAL;
  NOSEUSA: CHAR;

(* Inicio del programa INFLACION *)

BEGIN
  { Entrada de datos }
  WRITELN;
  WRITELN ( OUTPUT );
  WRITELN ( OUTPUT, 'POR FAVOR, indique la cantidad máxima de años:' );
  READLN ( MAXANOS );

  (* Inicialización de variables *)
  ANO      := 0;
  FACTOR1 := 1.0;
  FACTOR2 := 1.0;
  FACTOR3 := 1.0;

  { Cálculos y salida de datos }
  WRITELN ( OUTPUT );
  WRITELN ( ' ANO      7%      8%    10%' );
  WRITELN;
  REPEAT
    ANO      := ANO + 1;
    FACTOR1 := ( FACTOR1 * 1.07 );
```

```

FACTOR2 := ( FACTOR2 ) * 1.08;
FACTOR3 := FACTOR3 * ( 1.10 );

WRITELN ( ANO: 5, FACTOR1: 7:3, FACTOR2: 7:3 FACTOR3: 7:3 )

UNTIL ANO = MAXANOS;
WRITELN;
WRITELN ( OUTPUT )
END .

(* Fin del programa INFLACION *)

```

Cuya salida o resultados son los siguientes:

ANO	7%	8%	10%
1	1.070	1.080	1.100
2	1.145	1.166	1.210
3	1.225	1.260	1.331
4	1.311	1.360	1.464
5	1.403	1.469	1.611
6	1.501	1.587	1.772
7	1.606	1.714	1.949
8	1.718	1.851	2.144
9	1.838	1.999	2.358
10	1.967	2.159	2.594

la manera de ejecutar el compilador PAZCAL es la siguiente:

```
C:\> java -jar PAZCAL.jar INFLACION.PAZCAL
```

El compilador de PAZCAL debe abrir el archivo INFLACION.PAZCAL y empezar a leerlo línea por línea; cada una de esas líneas debe ser analizada sintáctica y semánticamente, a fin de detectar errores.

PAZCAL debe crear un nuevo archivo de salida de errores, con el mismo nombre del archivo en PAZCAL, solo que con el sufijo “-errores” y extensión TXT

Para el ejemplo citado, se generará el archivo INFLACION-errores.txt

Este archivo llevará, en primera instancia, una copia del programa en PAZCAL, con las líneas debidamente enumeradas hasta un máximo de 99,999 líneas (con ceros a la izquierda); se puede asumir que nunca ningún programa PAZCAL superará ese límite de líneas (99,999 o sea una menos que cien mil).



Entonces, se vería así:

INFLACION-errores.txt

```

00001 PROGRAM INFLACION ( INPUT, OUTPUT );
00002 {
00003     Asumiendo tasas de inflación anual de 7%, 8%, y 10%,
00004     encontrar el factor por el cual cualquier moneda, tales como

```

	<p style="text-align: center;">UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

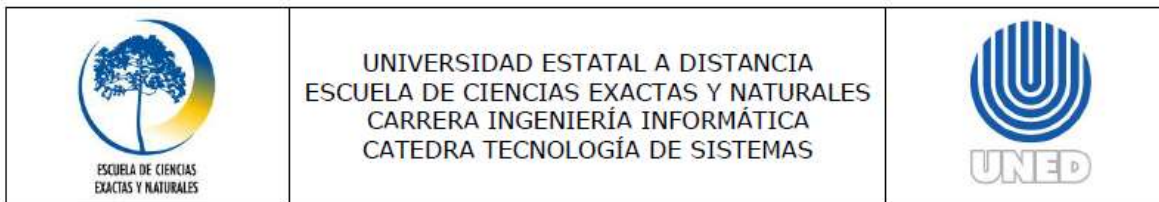
```

00005      el franco, el dólar, la libra esterlina, el marco, el rublo, el yen
00006      o el florín han sido devaluadas en 1, 2, ....., N anos.
00007  }
00008
00009  VAR
00010      MAXANOS: INTEGER;
00011      ANO:      INTEGER;
00012      FACTOR1: REAL;
00013      FACTOR2: REAL;
00014      FACTOR3: REAL;
00015      NOSEUSA: CHAR;
00016
00017  (* Inicio del programa INFLACION *)
00018
00019  BEGIN
00020      { Entrada de datos }
00021      WRITELN;
00022      WRITELN ( OUTPUT );
00023      WRITELN ( OUTPUT, 'POR FAVOR, indique la cantidad máxima de años:' );
00024      READLN ( MAXANOS );
00025
00026      (* Inicialización de variables *)
00027      ANO      := 0;
00028      FACTOR1  := 1.0;
00029      FACTOR2  := 1.0;
00030      FACTOR3  := 1.0;
00031
00032      { Cálculos y salida de datos }
00033      WRITELN ( OUTPUT );
00034      WRITELN ( ' ANO      7%      8%      10%' );
00035      WRITELN;
00036      REPEAT
00037          ANO      := ANO + 1;
00038          FACTOR1  := ( FACTOR1 * 1.07 );
00039          FACTOR2  := ( FACTOR2 ) * 1.08;
00040          FACTOR3  := FACTOR3 * ( 1.10 );
00041
00042          WRITELN ( ANO: 5, FACTOR1: 7:3, FACTOR2: 7:3 FACTOR3: 7:3 )
00043
00044      UNTIL ANO = MAXANOS;
00045      WRITELN;
00046      WRITELN ( OUTPUT )
00047  END .
00048
00049  (* Fin del programa INFLACION *)

```

NOTA: Es importante aclarar en este punto que el proyecto simula un compilador y que el objetivo principal es detectar TODOS los errores en el programa, en este sentido si el compilador encuentra un error deberá añadirlo al archivo de errores pero deberá continuar evaluando por más errores en la línea analizada y en las siguientes líneas.

Si no se encontraron errores quiere decir que el programa en PAZCAL es compatible con PASCAL y por lo tanto puede ser ejecutado en PASCAL.



Entonces el compilador PAZCAL debe invocar automáticamente al compilador PASCAL llamado **FPC.exe** para que el programa en PAZCAL sea corrido. Este programa se debe descargar desde la dirección antes citada (véase el segundo enlace citado arriba en las **PAUTAS**).

Para implementar este mecanismo el estudiante deberá investigar cómo manejar las variables de ambiente de CMD necesarias para realizar esas invocaciones, así como valorar si temporalmente debe cambiar la extensión del archivo **.PAZCAL** a **.pas** (porque **.pas** es la extensión de los archivos que maneja FREE PASCAL) o bien sacar una copia con extensión **.pas** para poder ser compilada en PASCAL y posteriormente ejecutada en PASCAL.

Si el programa no generó errores deberá crear un archivo **.PAS** directamente.

Este es un excelente artículo que explica el pase de parámetros desde CMD a un programa Java:

<https://personales.unican.es/corcuerp/java/Slides/CommandLineArguments.pdf>

En otras palabras, si el programa anterior en PAZCAL, INFLACION.PAZCAL, no tuviera errores, entonces así se vería la invocación del comando:

C:\FPC INFLACION.PAS

Este comando genera un archivo ejecutable de CMD con extensión EXE el cual simplemente luego se ejecuta en la línea de CMD; para el caso anterior sería así:

C:\INFLACION



Lo hacemos de esta manera porque programar el compilador PAZCAL es en sí, una tarea muy laboriosa y complicada; si a eso le agregáramos toda la lógica para generar un ejecutable real para la máquina la faena sería de nunca acabar. Aprovechando entonces la compatibilidad de PAZCAL con PASCAL invocamos el programa FREE PASCAL que es el que genera el ejecutable para el ambiente.

Tu compilador debe hacer la invocación del compilador FREE PASCAL y del ejecutable EXE de manera totalmente automática y transparente para el usuario, dando la ilusión de que es tu compilador PAZCAL quien lo está ejecutando.

Es importante destacar que el lenguaje de programación PAZCAL es un subconjunto del lenguaje de programación PASCAL, de tal manera que para cualquier duda que se tenga se puede consultar un manual técnico de PASCAL (véase el cuarto enlace citado arriba en las **PAUTAS**), ya que las instrucciones son las mismas, excepto que para PAZCAL vamos a precisar las siguientes reglas específicas:

Los archivos de código fuente en PAZCAL se pueden escribir en cualquier editor, siempre y cuando se grabe el contenido como "texto sin formato" (en código ASCII). Las líneas de estos archivos deben terminar con ENTER o RETURN.

1. El formato de la línea debe ser el siguiente y se debe validar de manera estricta:
 - a. Ninguna línea puede tener más de 150 caracteres.

	<p style="text-align: center;">UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

- b. Toda línea debe terminar con un punto y coma excepto en aquellos casos que expresamente se indique lo contrario en este enunciado.
- c. Los comentarios se delimitan por los caracteres { y } o por (* y *) y no terminan con punto y coma. Además:
 - 1. los comentarios pueden tomar más de una línea (eso se llama multilínea)
 - 2. en tal caso la primera línea debe empezar con { o (*
 - 3. y la última línea debe finalizar con } o *) respectivamente;
 - 4. esto no se puede asumir y se debe validar que todo comentario incluya ambos caracteres { y } o (* y *);
 - 5. los comentarios deben empezar y terminar de manera correspondiente, o sea que si empieza con { debe terminar con }; y si empieza con (* debe terminar con *);
 - 6. Todo lo que venga dentro del comentario, o sea dentro de { y } o dentro de (* y *) no se debe compilar.
- d. Se permiten líneas en blanco con 0, 1, 2, 3 y hasta 150 blancos, las cuales se deben ignorar y no se deben compilar por lo tanto se deben permitir líneas que tengan un blanco, o dos, o tres, o así, hasta 150 blancos.
- e. Las líneas en blanco pueden aparecer al principio del programa (antes de la instrucción PROGRAM), al final (luego de la instrucción END que finaliza el programa) o bien intercaladas entre otras líneas del programa.
- f. Decimos que el carácter blanco es el que corresponde a la espaciadora; se puede asumir que otros caracteres similares como tabuladores (TAB) o caracteres invisibles "que parecen blancos" no vendrán en el programa en PAZCAL.
- g. De la misma manera, se podrá asumir que los caracteres propios del idioma español como ¿, !, tildes, eñes, etc. no vendrán en el programa en PAZCAL ni siquiera en los comentarios (para no meter ruido).
- h. Los blancos de más se deben procesar como si fueran uno solo y por lo tanto se ignoran; en ese sentido no existen reglas: los lexemas pueden venir precedidos o seguidos por cero o más blancos, incluso pueden venir blancos luego de los puntos y coma al final de una línea o luego de que se cierra un comentario. Todos estos blancos, llamados "superfluos", en cualquier parte de la línea, simplemente se deben ignorar.

Los archivos de código fuente en PAZCAL siempre tendrán la siguiente estructura sin excepción; no se debe asumir y se debe verificar; las secciones deben aparecer en el orden que se indica a continuación:

PROGRAM NombreDePrograma (INPUT, OUTPUT);

Sección en donde se identifica al programa.

En donde:

NombreDePrograma: es un identificador de hasta 15 caracteres, de solo letras.

INPUT: es una palabra reservada; denota que habrá entrada de datos; es opcional.



OUTPUT: es una palabra reservada; denota que habrá salida de datos; es opcional.

Si aparecen ambas palabras reservadas se separan por coma y deben aparecer en ese orden.

Si aparece solo una palabra reservada no debe aparecer la coma.

Si no aparecen ninguna de las dos palabras reservadas no deben aparecer los paréntesis ni la coma y la línea debe terminar con punto y coma.

El hecho de que sean opcionales se debe a que son los valores por defecto ("default") de los dispositivos de entrada y salida de datos.

	<p>UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

VAR

Sección donde se definen las variables que utiliza el programa; **es opcional**.

Si aparece debe venir al menos una declaración de variable.

Si no aparece entonces no pueden venir declaraciones de variables.

Se explica en detalle más adelante.

BEGIN

END

Sección donde se indican los comandos que utiliza el programa; es obligatoria.

Y debe aparecer en ese orden: primero BEGIN, luego al menos un comando y finalmente END.

No pueden venir comandos antes del BEGIN o luego del END, excepto los comentarios como ya se explicó.

La línea que lleva BEGIN no debe terminar con punto y coma.

La línea que lleva END no debe terminar con punto y coma.

El programa como tal, en la última línea, debe terminar con punto luego del END.

Además:

Cualquier comando que no se reconozca como válido de PAZCAL (pero sí de PASCAL ver lista en anexo) se ignorará, pero en el archivo de errores saldrá el mensaje:

Advertencia: instrucción xxxx no es soportada por esta versión.

Estas advertencias no se considerarán errores.

Por ejemplo: las instrucciones "WHILE" y "FOR".

Una vez detectados estos comandos que PAZCAL no soporta el resto de la línea se debe ignorar y por lo tanto no compilar; la idea es permitir instrucciones no soportadas por PAZCAL pero sí por PASCAL. Para poder implementar esto, se sugiere llevar un arreglo o vector de todas las palabras reservadas de PASCAL (el alumno, si gusta, puede hacerlo de otra manera que juzgue conveniente; al final de este documento se anexan todas las palabras reservadas de PASCAL). Para los comandos que no estén en esta lista y que no correspondan a variables o instrucciones de PAZCAL deberá aparecer un mensaje como el siguiente:

ERROR: "xxxx" no es una instrucción válida de PAZCAL ni de PASCAL.
(estos errores sí se considerarán como tales).

MAYÚSCULAS/MINÚSCULAS

En general, PAZCAL no es sensible a mayúsculas ni minúsculas, en cuanto a los elementos del lenguaje. Sin embargo, lo que vaya dentro de las comillas simples se debe respetar literalmente.

Ejemplos:

Estos comandos son el mismo: WRITELN, writeln, WRITEln, writeLN, WrltEIN

Estas variables son la misma: FACTOR1 factor1 Factor1 FACTor1

COMENTARIOS

Como ya se indicó, los comentarios deben empezar con { o (* y terminar con } o *), respectivamente.

Todo lo que venga dentro de esos caracteres se debe ignorar y no compilar.

Ejemplos:

```
{
    Asumiendo tasas de inflación anual de 7%, 8%, y 10%,
    encontrar el factor por el cual cualquier moneda, tales como
    el franco, el dólar, la libra esterlina, el marco, el rublo, el yen
```


o el florín han sido devaluadas en 1, 2,, N años.
 }

(* Inicio del programa INFLACION *)

{ Cálculos y salida de datos }

UN COMANDO POR LÍNEA

PAZCAL solo permite un comando por línea; se puede asumir que esta regla siempre se cumplirá. Y siempre debe terminar con un punto y coma, excepto en aquellos casos que expresamente se indique lo contrario en este enunciado.

Ejemplos:

Correcto:

```
VAR
  MAXANOS: INTEGER;
  ANO:     INTEGER;
  FACTOR1: REAL;
  FACTOR2: REAL;
  FACTOR3: REAL;
  NOSEUSA: CHAR;

(* Inicio del programa INFLACION *)

BEGIN
  { Entrada de datos }
  WRITELN;
  WRITELN ( OUTPUT );
  WRITELN ( OUTPUT, 'POR FAVOR, indique la cantidad máxima de años:' );
  READLN ( MAXANOS );
```

Incorrecto:

```
VAR
  MAXANOS: INTEGER;      ANO:      INTEGER;      <--- NO.
  FACTOR1: REAL;
  FACTOR2: REAL;
  FACTOR3: REAL;        NOSEUSA: CHAR;          <--- NO.

(* Inicio del programa INFLACION *)

BEGIN
  { Entrada de datos }
  WRITELN; WRITELN ( OUTPUT );                                <--- NO.
  WRITELN ( OUTPUT, 'POR FAVOR, indique la cantidad máxima de años:' );
  READLN ( MAXANOS );
```

AGRUPACIÓN

Los caracteres de agrupación válidos para PAZCAL son:

(paréntesis izquierdo
) paréntesis derecho

Ejemplos:

```
FACTOR1 := ( FACTOR1 * 1.07 );
```

FACTOR2 := (FACTOR2) * 1.08;

FACTOR3 := FACTOR3 * (1.10);

Se debe reportar como error la falta de paréntesis de apertura, de paréntesis de cierre o ambos, así como los paréntesis de menos o de más.

Se usarán tantos paréntesis como se juzgue necesario para darle claridad a las instrucciones

Se recomienda investigar sobre las notaciones prefija, infija y postfija así como el uso de pilas para implementar algoritmos de evaluación de expresiones (como éstas que aparecen a la derecha del :=) o bien de condiciones (como las que aparecen en la cláusula UNTIL del comando REPEAT que se expone más adelante).

Un excelente artículo al respecto se puede consultar en este enlace:

<file:///C:/Users/GB-LTDES/Downloads/2054-Texto%20del%20art%C3%ADculo-6100-1-10-20141110.pdf>

CONSTANTES

En PAZCAL hay dos tipos de constantes: de texto y numéricas (que pueden ser enteras o reales).

Las de texto son de un solo carácter y van delimitadas por comillas simples.

Lo que va dentro de las comillas simples debe respetarse en cuanto a minúsculas y mayúsculas.

Ejemplos:

'S'

'N'

's'

'n'

Las numéricas pueden ser positivas o negativas, llevar decimales (el separador de decimales es el punto) y no deben llevar comas.

Hay 2 tipos de constantes numéricas:

enteras: entre -32768 y +32767 inclusive.

Ejemplos:

1

-1

10

-10

reales: números reales positivos o negativos que tiene punto decimal y al menos un dígito decimal. Entre -100000000 y +100000000 (cien millones) inclusive.

Ejemplos:

1.0

-1.0

10.23



-10.23

Nótese que 1. y 10. son incorrectas porque deben tener al menos un dígito decimal.

OPERADORES ARITMÉTICOS

Los siguientes son los operadores aritméticos que maneja PAZCAL:

Precedencia	Operador	Operación	Ejemplo	En PAZCAL
1	*	Multiplicación	XY/Z	$X * Y / Z$

	<p style="text-align: center;">UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

1	/	División	$X+Y/Z$	$X + (Y / Z)$
2	+	Suma	$X+Y/Z$	$(X + Y) / Z$
2	-	Resta	$X-Z/Y$	$(X - Y) / Z$

Las operaciones entre paréntesis se ejecutan primero.

Nótese de los ejemplos anteriores y de sus equivalentes en PAZCAL el uso adecuado de los paréntesis para cambiar las reglas de precedencia de los operadores aritméticos; cuando dos de ellos tienen la misma precedencia entonces se hacen los cálculos de izquierda a derecha.

OPERADORES RELACIONALES

Los siguientes son los operadores relacionales que maneja PAZCAL y que aparecen en las condiciones de la cláusula UNTIL del comando REPEAT expuesto más adelante:

= Igual.
<> Diferente.
< Menor que.
> Mayor que.
<= Menor o igual.
>= Mayor o igual.

INICIO/FIN DE COMANDOS

Como ya se indicó, esta sección empieza con la palabra reservada BEGIN (sin punto y coma al final) y termina con la palabra reservada END (sin punto y coma al final) seguida por el punto final del programa.

Ejemplo:



```
BEGIN
  { Entrada de datos }
  WRITELN;
  WRITELN ( OUTPUT );
  WRITELN ( OUTPUT, 'POR FAVOR, indique la cantidad maxima de anos:' );
  READLN ( MAXANOS );

  (* Inicialización de variables *)
  ANO      := 0;
  FACTOR1  := 1.0;
  FACTOR2  := 1.0;
  FACTOR3  := 1.0;

  { Calculos y salida de datos }
  WRITELN ( OUTPUT );
  WRITELN ( ' ANO      7%      8%    10%' );
  WRITELN;
  REPEAT
    ANO      := ANO + 1;
    FACTOR1  := ( FACTOR1 * 1.07 );
    FACTOR2  := ( FACTOR2 ) * 1.08;
    FACTOR3  := FACTOR3 * ( 1.10 );

    WRITELN ( ANO: 5, FACTOR1: 7:3, FACTOR2: 7:3 FACTOR3: 7:3 )

  UNTIL ANO = MAXANOS;
  WRITELN;
  WRITELN ( OUTPUT )
END .
```

	<p>UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

Un detalle importante: el último comando antes de END no debe terminar con punto y coma.

PALABRAS RESERVADAS

Los identificadores que correspondan a los comandos o instrucciones que usa PAZCAL o PASCAL decimos que son palabras reservadas y por lo tanto no pueden ser identificadores de variables. Como ya se indicó, para poder implementar esto se deberá llevar un arreglo o vector de todas las palabras reservadas de PASCAL, o bien implementar una solución que a criterio del estudiante permita llevar este control. Al final de este documento se anexan todas las palabras reservadas de PASCAL.

Ejemplos:

CONST
FOR
WHILE
IF
RECORD
TYPE

IDENTIFICADORES O NOMBRES DE VARIABLES

En PAZCAL los identificadores se definen siguiendo las siguientes reglas:

Todos los identificadores deben de comenzar con una letra.

Se permiten hasta 15 caracteres de longitud.

Se pueden usar letras o dígitos.

No son significativas las mayúsculas o minúsculas.

No se pueden utilizar las palabras reservadas como identificadores.

No se pueden usar caracteres especiales.

Ejemplos:

MAXANOS
ANO
FACTOR1
FACTOR2
FACTOR3
NOSEUSA

DEFINICIÓN DE VARIABLES

En PAZCAL las variables se definen inmediatamente después de la declaración del programa y dentro de la sección respectiva que viene delimitada por la palabra reservada VAR, la cual debe aparecer sola en esa línea y no debe llevar punto y coma al final. Esta sección siempre debe aparecer antes del BEGIN que denota el inicio del programa.

Ejemplo:

```
VAR
  MAXANOS: INTEGER;
  ANO:     INTEGER;
  FACTOR1: REAL;
  FACTOR2: REAL;
  FACTOR3: REAL;
  NOSEUSA: CHAR;
```

Las variables se declaran de la siguiente manera: **Identificador : tipo-dato;** en dónde:

Identificador se refiere a uno que siga las pautas indicadas líneas arriba;
 los dos puntos se utilizan para preceder el tipo de datos;
 el tipo-dato puede ser CHAR, INTEGER o REAL.

CHAR: almacena un (1) caracter;

INTEGER: almacena un número entero;

REAL: almacena un número real;

en los 3 casos se les puede asignar ya sea una constante, una expresión u otra variable del mismo tipo de datos.

debe validarse que las constantes que se asignen a las variables estén dentro de los límites establecidos según el tipo de datos para el que se haya definido.

Ejemplos:

Correctos:

```
ANO      := 0;
FACTOR1  := 1.0;
FACTOR2  := 1.0;
FACTOR3  := 1.0;
NOSEUSA  := 'N';
```

Incorrectos:

```
ANO      := 1.0;
MAXANOS  := 1.;
FACTOR1  := 1,0;
FACTOR2  := 100000001;
FACTOR3  := 1A;
NOSEUSA  := 'NO';
```

la declaración de una variable siempre debe terminar con punto y coma.

En cuanto a la conversión de tipos de datos:

A una variable se le puede asignar una constante, otra variable o una expresión del mismo tipo de datos. La otra variable o las variables contenidas en la expresión deben haberse definido previamente.

Tome en cuenta que si se dividen dos enteros pueden generar un real, por lo que deberá validar que el dato final podría ser de otro tipo.

ERRORES

PAZCAL debe reportar al usuario los errores que detecta cuando analiza las líneas de caracteres que conformar el archivo fuente que corresponde al programa.

Los diversos mensajes de error deben tener este formato:

ERROR 999: texto del error.

Todos los errores que se vayan a manejar deben ser identificados por un código y un texto. La enumeración de los errores queda a criterio del estudiante. Los errores deben ser claros y concisos y referirse a solo una situación de error por vez, de manera que un texto como éste:

Variable no definida o de tipo inválido.

no es correcto pues son dos errores diferentes en un mismo mensaje.

Ejemplos:

ERROR 025: tipos de datos no son compatibles.

ERROR 027: identificador no definido.

ERROR 030: falta un paréntesis derecho.

Los errores deben aparecer a partir de la columna 7 de la línea respectiva en el archivo de errores, de manera que así se facilite asociar, visualmente, los errores a la línea del programa PAZCAL respectivo; no olvidar que una sola línea puede tener varios errores y que por lo tanto todos se deben mostrar conforme a lo que se acaba de indicar, siempre hacia abajo y **NO HACIA LA DERECHA**.

También es importante recalcar, una vez más, que en el archivo de errores deben aparecer **todas las líneas del programa PAZCAL, tanto las válidas como las inválidas**, debidamente enumeradas a 5 dígitos (con ceros a la izquierda) y con los mensajes de error asociados (si los hay) abajo, uno por línea, tal y como se acaba de explicar. Esto es IMPORTANTÍSIMO para la revisión de las tareas y de la entrega final del proyecto.

Ejemplo:

```

00001 PROGRAM INFLACION ( INPUT, OUTPUT );
00002 {
00003     Asumiendo tasas de inflacion anual de 7%, 8%, y 10%,
00004     encontrar el factor por el cual cualquier moneda, tales como
00005     el franco, el dolar, la libra esterlina, el marco, el rublo, el yen
00006     o el florín han sido devaluadas en 1, 2, ...., N anos.
00007 }
00008
00009 VAR
00010     MAXANOS% INTEGER.
    ERROR 46: MAXANOS% no es un identificador valido.
    ERROR 80: INTEGER es reservada; no se puede usar como identificador.
00011     ANO: INTEGER;
00012     FACTOR1: REAL;
00013     FACTOR2: REAL;
00014     FACTOR3: REAL;
00015     NOSEUSA: CHAR;
  
```

ASIGNACIÓN DE VARIABLES

La asignación de valores a las variables se hace mediante `:=`, los cuales deben venir juntos sin blancos entre ellos. Lo que viene a la derecha se conoce como una expresión, la cual puede ser algo tan sencillo como una constante, otra variable o una fórmula matemática escrita con los elementos del lenguaje PAZCAL que se han expuesto en este documento. Se debe validar que la fórmula sea correcta no solo en cuanto a sintaxis (por ejemplo: que no falte un paréntesis de apertura o uno de cierre) sino también en cuanto a semántica (por ejemplo: que no se use un identificador que no se haya definido o que alguna de los identificadores presentes en la expresión no sean del tipo de datos de la variable que está al lado izquierdo del `:=`).

Un excelente artículo al respecto se puede consultar en este enlace:



<file:///C:/Users/GB-LTDES/Downloads/2054-Texto%20del%20art%C3%ADculo-6100-1-10-20141110.pdf>

COMANDOS

Se describen a continuación cada uno de los comandos que maneja PAZCAL.

READLN

Sintaxis: READLN (INPUT, VARIABLE)

	<p>UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

Permite ingresar un valor a una variable desde el teclado.

Se puede asumir que solo se pide una variable a la vez.

La palabra INPUT es opcional; si no aparece tampoco debe aparecer la coma.

La variable debe haber sido definida previamente.

Ejemplo:

```
READLN ( MAXANOS );
```

WRITELN

Sintaxis: WRITELN (OUTPUT, 'Texto', Variable: ANCHO:DECIMALES)

Para imprimir en pantalla un texto, una variable, ambos o ninguno.

La palabra OUTPUT es opcional; si no aparece tampoco debe aparecer la coma siguiente.

El texto es opcional; debe ir delimitado por comillas simples; si no aparece tampoco debe aparecer la coma siguiente.

La variable es opcional.

La variable debe haber sido definida previamente.

Se debe usar la coma como el separador de texto y variable (si hay ambas).

Si no hay ninguna de las dos entonces equivale a imprimir una línea en blanco.

Para cada variable que se vaya a imprimir se deben indicar los dos puntos y luego dos números enteros, a saber:

ANCHO: que se refiere a la cantidad de caracteres de impresión que tomará el dato contenido en la variable; siempre se debe indicar.

DECIMALES: que se refiere a la cantidad de caracteres de impresión que tomará el dato contenido en la variable cuando la misma es de tipo REAL; es obligatorio si el tipo de datos es REAL; si el tipo de datos es otro entonces no debe aparecer.

Ejemplos:

```
WRITELN;  
WRITELN ( OUTPUT );  
WRITELN ( OUTPUT, 'POR FAVOR, indique la cantidad máxima de años:' );  
WRITELN ( AÑO: 5, FACTOR1: 7:3, FACTOR2: 7:3 FACTOR3: 7:3 )
```

REPEAT

Sintaxis: REPEAT

comandos

UNTIL condición;

Para hacer un ciclo.

La palabra REPEAT debe aparecer en una sola línea; no debe llevar punto y coma al final.

Los comandos dentro del ciclo pueden ser cualquiera de los citados en esta sección

o bien cualquier otro que sea propio de PASCAL tal y como ya se explicó.

Sin embargo, no se permiten REPEAT dentro de otro REPEAT; se puede asumir que eso no pasará.

El último comando antes de UNTIL no debe finalizar con punto y coma.

La palabra UNTIL debe aparecer como última luego del último comando dentro del ciclo

y debe finalizar con una condición, la cual se puede asumir, si está presente, que siempre vendrá en la misma línea de UNTIL; se debe validar su presencia y correctitud.

La condición podrá estar formada por una expresión booleana que contendrá:

paréntesis

variables

operadores relacionales

constantes

Se debe validar la correctitud de esta expresión no solo en cuanto a sintaxis, por ejemplo:

- que todos los paréntesis de apertura tengan su correspondiente de cierre;
- que todos los paréntesis estén en posiciones válidas;
- que las variables existan;
- que el operador relacional sea válido;
- que los dos lados de la expresión sean de los mismos tipos de datos;

entre otras validaciones; sino también en cuanto a semántica:

- que las variables usadas en las expresiones existan;
- que los dos lados de la expresión sean de los mismos tipos de datos;

entre otras validaciones.

Un excelente artículo al respecto se puede consultar en este enlace:

<file:///C:/Users/GB-LTDES/Downloads/2054-Texto%20del%20art%C3%ADculo-6100-1-10-20141110.pdf>

Ejemplo:

```
REPEAT
  ANO      := ANO + 1;
  FACTOR1  := ( FACTOR1 * 1.07 );
  FACTOR2  := ( FACTOR2 ) * 1.08;
  FACTOR3  := FACTOR3 * ( 1.10 );

  WRITELN ( ANO: 5, FACTOR1: 7:3, FACTOR2: 7:3 FACTOR3: 7:3 )

UNTIL ANO = MAXANOS;
```

IMPORTANTE:

Si para cualquiera de las instrucciones de PAZCAL después de cumplir con la sintaxis respectiva, se detecta que viene otro lexema que corresponde a una palabra reservada de PASCAL (por ejemplo: ABSOLUTE) entonces se puede asumir que desde esa palabra reservada y hasta el punto y coma final de la línea la sintaxis es correcta; esto se hace porque la idea es permitir otras cláusulas de PASCAL dentro de una instrucción de PAZCAL; en otras palabras, su programa solo debe validar la sintaxis indicada en este enunciado, el resto lo puede ignorar siempre y cuando empiece con una palabra reservada válida de PASCAL.

Ejemplo:

```
VAR
  CODIGO1 : INTEGER;
  CODIGO6 : INTEGER ABSOLUTE CODIGO1;
```

ARCHIVO PARA PRUEBAS

Como ya mostramos al principio, un ejemplo de programa en PAZCAL es el que aparece en el apartado **"EJEMPLO DE USO DEL COMPILADOR!"** y sobre el cual hemos basado todo este enunciado. El problema es que en la vida real los programas no vienen tan "bonitos": bien identados, claramente escritos y acomodados; en realidad, por experiencia propia, como todos sabemos, los programadores somos descuidados al respecto, así que a continuación mostramos el mismo programa pero "feo", en el sentido de que no es muy ordenado; es importante que las pruebas de tu compilador las hagas usando este programa "feo", ya que así serán los archivos de prueba que usará el Profesor. Toma nota, por favor, de que las primeras líneas tienen desde 0 y hasta muchos caracteres en blanco, antes de que aparezca PROGRAM; los mismo las líneas intermedias y las finales luego de END; además, hay blancos superfluos por todo lado: antes y después de lexemas; todo eso tu programa deber permitirlo e ignorarlo siempre y cuando no sean errores:



UNIVERSIDAD ESTATAL A DISTANCIA
ESCUELA DE CIENCIAS EXACTAS Y NATURALES
CARRERA INGENIERÍA INFORMÁTICA
CATEDRA TECNOLOGÍA DE SISTEMAS



INFLACION.PAZCAL (versión "fea"); ojo a las primeras líneas en blanco

```
PROGRAM      INFLACION      (INPUT,OUTPUT      )      ;

{
    Asumiendo tasas de inflacion anual de 7%, 8%, y 10%,
    encontrar el factor por el cual cualquier moneda, tales como
    el franco, el dolar, la libra esterlina, el marco, el rublo, el yen
    o el florín han sido devaluadas en 1, 2, ....,      N anos.
}

VAR
MAXANOS      :      INTEGER      ;
ANO:      INTEGER;
FACTOR1      :      REAL      ;
FACTOR2: REAL;
FACTOR3      :      REAL;
NOSEUSA:      CHAR      ;

(* Inicio del programa INFLACION *)



BEGIN
    { Entrada de datos }
    WRITELN      ;
    WRITELN(OUTPUT);
    WRITELN ( OUTPUT , 'POR FAVOR, indique la cantidad máxima de años:' ) ;
    READLN(MAXANOS);

    (* Inicialización de variables *)
    ANO      :=      0      ;
    FACTOR1      :=      1.0      ;
    FACTOR2      :=      1.0      ;
    FACTOR3      :=      1.0      ;

    { Calculos y salida de datos }
    WRITELN ( OUTPUT ) ;
    WRITELN(' ANO      7%      8%      10%' );
    WRITELN;

    REPEAT
        ANO      :=      ANO      +      1      ;
        FACTOR1 :=( FACTOR1 * 1.07 );
        FACTOR2 := (FACTOR2 ) * 1.08      ;
        FACTOR3 :=      FACTOR3      *      ( 1.10      );

    WRITELN( ANO:5, FACTOR1: 7: 3, FACTOR2: 7 : 3 FACTOR3: 7:3)
```

	<p>UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
---	--	---

```

UNTIL      ANO      =      MAXANOS;

WRITELN;
      WRITELN ( OUTPUT )
END.

(* Fin del programa INFLACION *)

```

OJO: fíjate que hay líneas en blanco luego del último comentario.

ENTREGA

Se deberá entregar como proyecto en la entrega final previa a la defensa:

Todo el proyecto tal y como se organizó en NETBEANS, todas las carpetas, en particular los programas fuentes finales en Java. SUGERENCIA: entregar un ZIP con todo. De este ZIP el Profesor tomará el archivo .jar para probar el compilador; y abrirá los archivos .java para analizar y revisar la programación. **Si no viene todo se dará como no entregado.**

Un manual técnico que explique cómo instalar y ejecutar el compilador PAZCAL desde CMD (deben ser claras las instrucciones).



SE REITERA: MUY IMPORTANTE: la ejecución debe ser desde CMD, ya que el Profesor no entrará a Netbeans para probarlo. La mejor manera de estar seguros de que se ejecute bien, es ir a otra micro y ejecutarlo ahí, asegurándose que no dé problemas. **No se permite usar nombres de archivos fijos o "quemados"**. Si no se siguen estas recomendaciones y el programa no corre como se ha indicado entonces lamentablemente no se podrá calificar y se pondrá la nota cero.

Sin embargo, el Profesor analizará y revisará el código fuente del programa para verificar: **plagio**, uso de buenas técnicas de programación, indentación, identificadores significativos, modularidad, orden, etc.

En el manual técnico deberá venir una lista de los puntos que no se programaron o que no funcionan correctamente y la justificación del caso, a fin de valorar la calidad de la solución entregada y la viabilidad de dar por buenos algunos de esos puntos sin terminar.

DEFENSA

Una vez entregado el proyecto, se programará una cita con el Tutor para hacer una exposición virtual del proyecto y defenderlo mediante TEAMS, de manera individual. La revisión se basará en la rúbrica, al momento de asignar la nota, por qué le dio esa calificación. En general, sin embargo, el tutor verificará el uso de buenas técnicas de programación, cumplimiento de los objetivos del proyecto y plagio (**al respecto, se advierte que el Profesor usará herramientas de software para confirmar qué tan parecidos son los programas que entregan los alumnos y en caso de detectar plagio se realizara el debido proceso con la Cátedra, conforme a las normas vigentes**); se harán preguntas técnicas sobre el código fuente que el estudiante deberá explicar de manera convincente para demostrar que es el autor del programa.

 <p>ESCUELA DE CIENCIAS EXACTAS Y NATURALES</p>	<p>UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	
--	--	---

ANEXO:

Lista de palabras reservadas de PASCAL

ABS
 ABSOLUTE
 ABSTRACT
 ALIAS
 AND
 AND
 ARCTAN
 ARRAY
 AS
 AS
 ASM
 ASSEMBLER
 BEGIN
 BEGIN
 BITPACKED
 BOOLEAN
 BOOLEAN16
 BOOLEAN32
 BOOLEAN64
 BREAK
 BYTE
 BYTEBOOL
 CARDINAL
 CASE
 CDECL
 CHAR
 CHR
 CLASS
 COMP
 CONST
 CONST
 CONSTRUCTOR
 CONTINUE
 CPPCLASS
 CPPDECL
 CURRENCY
 CVAR
 DEFAULT
 DEPRECATED
 DESTRUCTOR
 DISPINTERFACE
 DISPOSE
 DIV
 DIV
 DO
 DOUBLE
 DOWNT0



UNIVERSIDAD ESTATAL A DISTANCIA
ESCUELA DE CIENCIAS EXACTAS Y NATURALES
CARRERA INGENIERÍA INFORMÁTICA
CATEDRA TECNOLOGÍA DE SISTEMAS



DYNAMIC
ELSE
END
END
ENUMERATOR
EOF
EOLN
EXCEPT
EXCLUDE
EXIT
EXP
EXPERIMENTAL
EXPORT
EXPORTS
EXTENDED
EXTERNAL
FAIL
FALSE
FAR
FAR16
FILE
FINALIZATION
FINALLY
FOR
FORWARD
FUNCTION
GENERIC
GET
GOTO
HELPER
IF
IMPLEMENTATION
IMPLEMENTS
IN
INCLUDE
INDEX
INHERITED
INITIALIZATION
INLINE
INPUT
INT64
INTEGER
INTERFACE
INTERRUPT
IOCHECK
IS
IS
LABEL
LIBRARY
LN
LOCAL



UNIVERSIDAD ESTATAL A DISTANCIA
ESCUELA DE CIENCIAS EXACTAS Y NATURALES
CARRERA INGENIERÍA INFORMÁTICA
CATEDRA TECNOLOGÍA DE SISTEMAS



LONGBOOL
LONGINT
LONGWORD
MESSAGE
MOD
MOD
NAME
NEAR
NEW
NIL
NODEFAULT
NORETURN
NOSTACKFRAME
NOT
NOT
OBJECT
ODD
OF
OLDFPCCALL
ON
OPERATOR
OR
OR
ORD
OTHERWISE
OUT
OUTPUT
OVERLOAD
OVERRIDE
PACKED
PAEK
PAGE
PASCAL
PLATFORM
POPSTACK
PRED
PRIVATE
PROCEDURE
PROGRAM
PROGRAM
PROPERTY
PROTECTED
PUBLIC
PUBLISHED
PUT
QWORD
QWORDBOOL
RAISE
READ
READ
READLN



UNIVERSIDAD ESTATAL A DISTANCIA
ESCUELA DE CIENCIAS EXACTAS Y NATURALES
CARRERA INGENIERÍA INFORMÁTICA
CATEDRA TECNOLOGÍA DE SISTEMAS



REAL
RECORD
REGISTER
REINTRODUCE
REPEAT
REPEAT
RESET
RESOURCESTRING
RESULT
REWRITE
SAFECALL
SAVEREGISTERS
SELF
SET
SHL
SHL
SHORTINT
SHR
SHR
SIN
SINGLE
SMALLINT
SOFTFLOAT
SPECIALIZE
SQR
SQRT
STATIC
STDCALL
STORED
STRICT
STRING
SUCC
THEN
THREADVAR
TO
TRUE
TRUNC
TRY
TRY
TYPE
UNALIGNED
UNIMPLEMENTED
UNIT
UNPACK
UNTIL
UNTIL
USES
VAR
VAR
VARARGS
VIRTUAL





UNIVERSIDAD ESTATAL A DISTANCIA
ESCUELA DE CIENCIAS EXACTAS Y NATURALES
CARRERA INGENIERÍA INFORMÁTICA
CATEDRA TECNOLOGÍA DE SISTEMAS



WHILE
WINAPI
WITH
WORD
WORDBOOL
WRITE
WRITELN
XOR

FIN.

 <p>ESCUELA DE CIENCIAS EXACTAS Y NATURALES</p>	<p>UNIVERSIDAD ESTATAL A DISTANCIA ESCUELA DE CIENCIAS EXACTAS Y NATURALES CARRERA INGENIERÍA INFORMÁTICA CATEDRA TECNOLOGÍA DE SISTEMAS</p>	 <p>UNED</p>
--	--	---

RÚBRICA DE CALIFICACIÓN

Criterio	Cumple a satisfacción lo indicado en la evaluación	Cumple medianamente en lo indicado en la evaluación	Cumple en contenido y formato pero los aportes no son significantes	No cumple o no presenta lo solicitado
1. Corrección de lo que no se cumplió de la Tarea1	25	12	6	0
2. Corrección de lo que no se cumplió de la Tarea2	15	7	3	0
3. Uso correcto del comando REPEAT: que aparezcan las dos palabras REPEAT y UNTIL, que REPEAT venga sola en una línea y sin punto y coma, que el comando antes de UNTIL no traiga punto y coma, que se valide que siempre venga condición (aunque ésta se revisa en el próximo punto) ,entre otras cosas a revisar sobre este tema.	25	12	6	0
4. Uso correcto del comando REPEAT: que se valide la correctitud de la condición que viene después del UNTIL y que termine o no con punto y coma según corresponda; además: que ambas expresiones a la izquierda y derecha del operador relacional sean válidas tanto sintáctica como semánticamente (en el enunciado se citan algunas de las	25	14	7	0



UNIVERSIDAD ESTATAL A DISTANCIA
 ESCUELA DE CIENCIAS EXACTAS Y NATURALES
 CARRERA INGENIERÍA INFORMÁTICA
 CATEDRA TECNOLOGÍA DE SISTEMAS



validaciones que se deben realizar), que se manejen los operadores aritméticos y relacionales válidos; manejo de constantes: tipos y tamaños adecuados según la definición de tipos del enunciado (char, integer, real); entre otras cosas a revisar sobre este tema.				
5. Creación del manual técnico	10	5	2	
TOTAL	100	70	40	0