PROYECTO VALOR 4%
DEFENSA VALOR 2%

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
KOVOL es una simulación básica, ficticia, del lenguaje de programación COBOL (**clásico**) 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, versión 8.2.
Debe ser programado en modo carácter. Si el estudiante desea hacerlo en modo gráfico no hay problema, pero no se brindan puntos adicionales en la nota final.

Se recomienda primero que todo empaparse un poco de COBOL, para lo cual es importante que lean estos enlaces a fin de ubicarse:

1. Historia del lenguaje COBOL y ubicación del contexto histórico
   https://hipertextual.com/2011/12/historia-de-la-tecnologia-el-lenguaje-cobol

   https://www.xataka.com/historia-tecnologica/legendario-lenguaje-programacion-cobol-acaba-cumplir-60-anos-probable-que-cumpla-otros-60

2. Usaremos como “ejecutador” el programa GnuCOBOL; aquí tenemos la guía del usuario:
   https://gnucobol.sourceforge.io/

3. Este es el enlace donde se puede descargar el GnuCOBOL:
   https://sourceforge.net/projects/gnucobol/files/

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., también tienen la libertad de tomar la iniciativa y buscar en el Internet material afín a los temas del curso.

PROYECTO KOVOL

NOCIONES BÁSICAS

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

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

La sintaxis para usar el compilador KOVOL es la siguiente, la cual se digita en la línea de comandos de MICROSOFT WINDOWS o CMD; este punto es importante porque el programa del estudiante, el profesor no lo va a probar desde Netbeans sino desde la consola CMD, de la siguiente forma:

C:\> java –jar KOVOL.jar ArchivoKOVOL.kovol

En donde ArchivoKOVOL.kovol es el nombre de archivo de texto con extensión .kovol 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 “kovol” como ya se indicó.
- Si no se indica la extensión se puede asumir.

Además, se debe entregar el proyecto y toda la solución de NetBeans, junto con todas las clases, ya que se le calificará que el código compile de forma correcta. Asegúrese de entregar toda la solución completa de forma que compile correctamente.

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 copiar el archivo *.jar, los archivos de entrada de la revisión y el 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 del compilador hacia la máquina y que se pueda ejecutar en cualquier parte.

EJEMPLO DE USO DEL COMPILADOR

Si tenemos el siguiente archivo de texto que realiza cálculos simples sobre dos números, tales como:
7 + 3 = 10
El cual se llama de la siguiente forma:

D:\CALCULOS.KOVOL

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

D:\CALCULOS.KOVOL

```
 IDENTIFICATION DIVISION.
  PROGRAM-ID. CALCULOS.
  AUTHOR. MARIO QUIROS.
*
*<-- Columna 7 con asterisco significa comentario
*
* CURSO COMPILADORES                    esta es la columna 72--> *
*
*8901<-- Este es el margen A (columnas 8 a 11)
*    2<-- Aqui empieza el margen B (columna 12)
*
 ENVIRONMENT DIVISION.

 DATA DIVISION.                                                  EstasSon
  WORKING-STORAGE SECTION.                                       columnas
   77 NUMERO1    PIC 9(2) VALUE ZEROS.                          73 A 80
   77 NUMERO2    PIC 9(2) VALUE ZEROS.
   77 RESULTADO1 PIC 9(2)V9(2) VALUE ZEROS.
   77 RESULTADO2 PIC 9(2)V9(2) VALUE ZEROS.
   77 RESULTADO3 PIC 9(2)V9(2) VALUE ZEROS.
   77 RESULTADO4 PIC S9(2)V9(2) VALUE ZEROS.
   77 RESULTADO5 PIC 9(2)V9(2) VALUE ZEROS.

 PROCEDURE DIVISION.
  INICIO.
*     <-- Aqui empieza el margen B (columna 12)
     DISPLAY "PRIMER  NUMERO: " WITH NO ADVANCING.
     ACCEPT NUMERO1.
     DISPLAY "SEGUNDO NUMERO: " WITH NO ADVANCING.
     ACCEPT NUMERO2.
     COMPUTE RESULTADO1 = NUMERO1 * NUMERO2.
     COMPUTE RESULTADO2 = NUMERO1 / NUMERO2.
     COMPUTE RESULTADO3 = NUMERO1 + NUMERO2.
     COMPUTE RESULTADO4 = NUMERO1 - NUMERO2.
     COMPUTE RESULTADO5 = NUMERO1 * NUMERO1 * (NUMERO2 * NUMERO2).
     DISPLAY "MULTIPLICACION: ", RESULTADO1.
     DISPLAY "DIVISION      : ", RESULTADO2.
     DISPLAY "SUMA          : ", RESULTADO3.
     DISPLAY "RESTA         : ", RESULTADO4.
     DISPLAY "EXPRESION     : ", RESULTADO5.
     STOP RUN.
```

La manera de ejecutar el compilador KOVOL es la siguiente:

C:\> java –jar KOVOL.jar D:\CALCULOS.kovol

El compilador de KOVOL debe abrir el archivo D:\CALCULOS.kovol y empezar a leerlo línea por línea; cada una de esas líneas debe ser analizada sintáctica y semánticamente, a fin de detectar errores.

KOVOL debe crear un nuevo archivo de salida de errores, con el mismo nombre del archivo original de KOVOL, solo que con el sufijo “-errores” y extensión *.txt

Para el ejemplo citado, se generará el archivo D:\CALCULOS-errores.txt

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

Entonces, se vería así:

D:\CALCULOS-errores.txt

```
00001  IDENTIFICATION DIVISION.
00002   PROGRAM-ID. CALCULOS.
00003   AUTHOR. MARIO QUIROS.
00004 *
00005 *<-- Columna 7 con asterisco significa comentario
00006 *
00007 * CURSO COMPILADORES                    esta es la columna 72--> *
00008 *
00009 *8901<-- Este es el margen A (columnas 8 a 11)
00010 *    2<-- Aqui empieza el margen B (columna 12)
00011 *
00012  ENVIRONMENT DIVISION.
00013
00014  DATA DIVISION.                                                EstasSon
00015   wORKING-STORAGE SECTION.                                     columnas
00016    77 NUMERO1    PIC 9(2) VALUE ZEROS.                         73 A 80
00017    77 NUMERO2    PIC 9(2) VALUE ZEROS.
00018    77 RESULTADO1 PIC 9(2)V9(2) VALUE ZEROS.
00019    77 RESULTADO2 PIC 9(2)V9(2) VALUE ZEROS.
00020    77 RESULTADO3 PIC 9(2)V9(2) VALUE ZEROS.
00021    77 RESULTADO4 PIC S9(2)V9(2) VALUE ZEROS.
00022    77 RESULTADO5 PIC 9(2)V9(2) VALUE ZEROS.
00023
00024  PROCEDURE DIVISION.
00025   INICIO.
00026 *     <-- Aqui empieza el margen B (columna 12)
00027       DISPLAY "PRIMER  NUMERO: " WITH NO ADVANCING.
00028       ACCEPT NUMERO1.
00029       DISPLAY "SEGUNDO NUMERO: " WITH NO ADVANCING.
00030       ACCEPT NUMERO2.
00031       COMPUTE RESULTADO1 = NUMERO1 * NUMERO2.
00032       COMPUTE RESULTADO2 = NUMERO1 / NUMERO2.
00033       COMPUTE RESULTADO3 = NUMERO1 + NUMERO2.
00034       COMPUTE RESULTADO4 = NUMERO1 - NUMERO2.
00035       COMPUTE RESULTADO5 = NUMERO1 * NUMERO1 * (NUMERO2 * NUMERO2).
00036       DISPLAY "MULTIPLICACION: ", RESULTADO1.
00037       DISPLAY "DIVISION      : ", RESULTADO2.
00038       DISPLAY "SUMA          : ", RESULTADO3.
00039       DISPLAY "RESTA         : ", RESULTADO4.
00040       DISPLAY "EXPRESION     : ", RESULTADO5.
00041       STOP RUN.
```

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

Entonces el compilador KOVOL debe invocar automáticamente al programa COBOL llamado **COBC.exe** para que el programa en KOVOL sea corrido. Este programa se debe descargar desde la dirección antes citada (véase el tercer 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 **.KOVOL** a **.cob** (porque .cob es la extensión de los archivos que maneja GnuCOBOL) o bien crear una copia con extensión **.cob** para poder ser compilada en COBOL y posteriormente ejecutada en COBOL.

En otras palabras, si el programa anterior en KOVOL, D:\CALCULOS.kovol, no tuviera errores, entonces la invocación del comando debe ser de la siguiente manera:

C:\cobc -x CALCULOS.kobol (esto es para compilar y generar el ejecutable o EXE)
C:\CALCULOS (esto es para ejecutar el programa ejecutable o EXE)

Lo hacemos de esta manera porque programar el compilador KOVOL 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 KOVOL con COBOL, invocamos el programa COBOL que es el que genera el ejecutable para el ambiente.

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

- Los archivos de código fuente en KOVOL 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.

- El formato de la línea debe ser el siguiente y se debe validar de manera estricta:
  - Las primeras 6 columnas deben venir en blanco.
  - La columna 7 debe venir en blanco o con un asterisco (en ese caso es un comentario) o con un guion (en ese caso es la continuación de una línea).
  - Las columnas de la 8 a la 11 se conocen como el margen A.
  - Las columnas de la 12 a la 72 se conocen como el margen B.
  - Las columnas de la 73 a la 80 son opcionales y son comentarios.
  - Ninguna línea puede tener más de 80 columnas.
  - Toda línea debe terminar con un punto.
  - Si no termina con punto entonces se deben leer las siguientes líneas hasta encontrar aquella que termine con punto; en ese caso se deben ignorar todas las líneas leídas desde la primera hasta la que lleva punto, asumir que son correctas, no compilarlas y proceder con la siguiente línea; la idea es permitir comandos COBOL de más de una línea de longitud.

- Además:
  - El punto, punto y coma o coma no pueden ir precedidos por un espacio, pero sí tienen que ir seguidos por un espacio.
  - Un paréntesis izquierdo no puede ir seguido por un espacio.
  - Un paréntesis derecho no puede ir precedido por un espacio.
  - Un operador aritmético o un signo igual tiene que ir precedido por un espacio y seguido por un espacio.
  - En cualquier otro contexto, los blancos de más se consideran como si fueran uno solo y por lo tanto se ignoran.

Los archivos de código fuente en KOVOL siempre tendrán la siguiente estructura sin excepción; no se debe asumir y se debe verificar; si no la cumplen se deben enviar los mensajes de error respectivos:

```
*Nótese los blancos delante de las líneas (en fondo negro).
 IDENTIFICATION DIVISION.
  PROGRAM-ID. Nombre-de-programa (hasta 30 caracteres, solo letras).
  AUTHOR. Nombre-de-autor (hasta 30 caracteres, solo letras).

 ENVIRONMENT DIVISION.

 DATA DIVISION.
  WORKING-STORAGE SECTION.

 PROCEDURE DIVISION.
  NOMBRES-PARRAFO-PROGRAMADOS.
     COMANDOS.
```

en donde:

- los nombres de división deben escribirse en el margen A (columnas 8 a 11);
- los nombres de párrafo (lo que va dentro de las divisiones) como PROGRAM-ID deben escribirse en el margen A (columnas 8 a 11);
- los nombres de sección (que van dentro de las divisiones y sirven para agrupar párrafos) deben escribirse en el margen A (columnas 8 a 11);
- los nombres de párrafo programados (siguiendo la nomenclatura de identificadores indicada más adelante) puestos por el programador (o sea el nombre del programa o "procedimientos") en la PROCEDURE DIVISION deben escribirse en el margen A (columnas 8 a 11); y
- el resto de los elementos del lenguaje (comandos) se deben escribir en el margen B (columnas 12 a 72) a no ser que expresamente se indique lo contrario.
- NOTA 1: dado que los nombres de división, sección y párrafos van en el margen A, se acostumbra indentarlos más o menos como se mostró en el ejemplo anterior, aunque en realidad todos

pueden empezar en la misma columna o en cualquiera de las 4, siempre y cuando estén dentro del margen A.

- NOTA2: las líneas en blanco son opcionales.

Físicamente, el archivo que almacena este código fuente se debe llamar `NombreArchivo.kobol` como ya se explicó. Es importante recordar que tanto en COBOL como en KOVOL los identificadores de archivo no son sensibles a mayúsculas y minúsculas.

- Cualquier comando que no se reconozca como válido de KOVOL (pero sí de COBOL) 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 "BACKGROUND-COLOR" e “INDEX”.

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

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

MAYÚSCULAS/MINÚSCULAS

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

Ejemplos:
Estos comandos son el mismo: DISPLAY display DISplay disPLAY
Estas variables son la misma: SALDO saldo Saldo saldO

COMENTARIOS

Como ya se indicó, un asterisco en la columna 7 de cualquier línea se debe interpretar como comentario y entonces se puede asumir que el resto de la línea es un comentario y por lo tanto no se debe compilar.
Además, también, como ya se indicó, lo que aparece en las columnas 73 a 80 de cada línea se considera un comentario.

IMPORTANTE:

Ambos tipos de comentarios se deben suprimir cuando se construya el archivo que se va a pasar a GnuCOBOL para su compilación, ya que no se soportan y dan error.

Ejemplos:
Lo que está en negrita son comentarios:

```
 IDENTIFICATION DIVISION.
  PROGRAM-ID. CALCULOS.
  AUTHOR. MARIO QUIROS.
*
*<-- Columna 7 con asterisco significa comentario
*
* CURSO COMPILADORES                    esta es la columna 72--> *
*
*8901<-- Este es el margen A (columnas 8 a 11)
*    2<-- Aqui empieza el margen B (columna 12)
*
 ENVIRONMENT DIVISION.

 DATA DIVISION.                                                  EstasSon
  WORKING-STORAGE SECTION.                                       columnas
   77 NUMERO1    PIC 9(2) VALUE ZEROS.                          73 A 80
```

UN COMANDO POR LÍNEA
KOVOL solo permite un comando por línea.
Y siempre debe terminar con un punto.
Si no termina con punto entonces se deben leer las siguientes líneas hasta encontrar aquella que termine con punto; en ese caso se deben ignorar todas las líneas leídas desde la primera hasta la que lleva punto, asumir que son correctas, no compilarlas y proceder con la siguiente línea; la idea es permitir comandos COBOL de más de una línea de longitud.
Se puede asumir que estas reglas siempre se cumplirán.

Ejemplos:

```
Correcto:
77 NUMERO1    PIC 9(2) VALUE ZEROS.
77 NUMERO2    PIC 9(2) VALUE ZEROS.
77 RESULTADO1 PIC 9(2)V9(2) VALUE ZEROS.
77 RESULTADO2 PIC 9(2)V9(2) VALUE ZEROS.

Incorrecto:
77 NUMERO1    PIC 9(2) VALUE ZEROS.      77 NUMERO2    PIC 9(2) VALUE ZEROS.
77 RESULTADO1 PIC 9(2)V9(2) VALUE ZEROS.
77 RESULTADO2 PIC 9(2)V9(2) VALUE ZEROS.
```

AGRUPACIÓN
Los caracteres de agrupación válidos para KOVOL son:

(paréntesis izquierdo
) paréntesis derecho

Ejemplo:

```
COMPUTE RESULTADO5 = NUMERO1 * NUMERO1 * (NUMERO2 * NUMERO2).
```

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 (véase el comando COMPUTE más adelante).

Se recuerda que:

- Un paréntesis izquierdo no puede ir seguido por un espacio.
- Un paréntesis derecho no puede ir precedido por un espacio.
- Un operador aritmético o un signo igual tiene que ir precedido por un espacio y seguido por un espacio.

CONSTANTES
En KOVOL hay tres tipos de constantes: de texto, numéricas y figurativas.

Las de texto tienen hasta 100 caracteres y van delimitadas por comillas simples.
Lo que va dentro de las comillas debe respetarse en cuanto a minúsculas y mayúsculas.

Ejemplos:
'HOLA'
'$25,000.00'
'Numero de empleados'

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
  Ejemplos:
  1
  -1
  10
  -10
- punto fijo: números reales positivos o negativos que tiene punto decimal y al menos un dígito decimal.
  Ejemplos:
  1.0
  -1.0
  10.23
  -10.23
  Nótese que 1. y 10. son incorrectas porque deben tener al menos un dígito decimal.

Las figurativas son las siguientes:

- ZERO (equivale al valor de cero).
- SPACE (equivale al valor de espacios necesario para llenar una variable de tipo texto).

OPERADORES ARITMÉTICOS
Los siguientes son los operadores aritméticos que maneja KOVOL:

Precedencia	Operador	Operación	Ejemplo	En KOVOL
1	*	Multiplicación	XY/Z	`X * Y / Z`
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 KOVOL 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.

Se recuerda que:

- Un paréntesis izquierdo no puede ir seguido por un espacio.
- Un paréntesis derecho no puede ir precedido por un espacio.
- Un operador aritmético o un signo igual tiene que ir precedido por un espacio y seguido por un espacio.

INICIO/FIN DE COMANDOS
En la PROCEDURE DIVISION los nombres de párrafo programados (siguiendo la nomenclatura de identificadores indicada más adelante) o sea puestos por el programador (y que corresponden al nombre del programa o "procedimientos") son aquellos que aparecen en el margen A (columnas 8 a 11) y pueden ser muchos, dependiendo de la modularidad con las que se haya hecho el programa en KOVOL.

A continuación aparecen, en el margen B los comandos de KOBOL o de COBOL.

Normalmente, los programas terminan con el comando STOP RUN que, como ya se indicó, debe ir en el margen B.

Ejemplo:
"Inicio" es un párrafo programado.

```
       PROCEDURE DIVISION.
        INICIO.
      *     <-- Aqui empieza el margen B (columna 12)
            DISPLAY "PRIMER  NUMERO: " WITH NO ADVANCING.
            ACCEPT NUMERO1.
            DISPLAY "SEGUNDO NUMERO: " WITH NO ADVANCING.
            ACCEPT NUMERO2.
            COMPUTE RESULTADO1 = NUMERO1 * NUMERO2.
            COMPUTE RESULTADO2 = NUMERO1 / NUMERO2.
            COMPUTE RESULTADO3 = NUMERO1 + NUMERO2.
            COMPUTE RESULTADO4 = NUMERO1 - NUMERO2.
            COMPUTE RESULTADO5 = NUMERO1 * NUMERO1 * (NUMERO2 * NUMERO2).
            DISPLAY "MULTIPLICACION: ", RESULTADO1.
            DISPLAY "DIVISION      : ", RESULTADO2.
            DISPLAY "SUMA          : ", RESULTADO3.
            DISPLAY "RESTA         : ", RESULTADO4.
            DISPLAY "EXPRESION     : ", RESULTADO5.
            STOP RUN.
```

PALABRAS RESERVADAS

Los identificadores que correspondan a los comandos o instrucciones que usa KOVOL o COBOL 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 COBOL, 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 COBOL.

Ejemplos:
FILLER
ADVANCING

NOMBRES DE VARIABLES

En KOVOL los identificadores se definen siguiendo las siguientes reglas:

- Todos los identificadores deben de comenzar con una letra.
- Se permiten hasta 30 caracteres de longitud.
- Se pueden usar letras, dígitos y el guion.
- No son significativas las letras mayúsculas y minúsculas.
- No pueden terminar con guion.
- No se pueden utilizar las palabras reservadas como identificadores.

DEFINICIÓN DE VARIABLES

En KOVOL las variables se definen mediante los niveles siguiendo la siguiente sintaxis:

```
NIVEL NOMBRE-VARIABLE PICT/PICTURE especificacion-formato VALUE constante.
```

en dónde:

- Nivel es “77” siempre; debe declararse en el margen A;
- NOMBRE-VARIABLE debe seguir las reglas citadas anteriormente para identificadores; debe declararse en el margen B;

- PICT o PICTURE (cualquiera de las dos) sirve para indicar el tipo de dato; debe declararse en el margen B;
- "especificacion-formato" es la definición del tipo de datos y tamaño de la variable, conforme a las siguientes pautas (debe declararse en el margen B):

    - Numéricas:
        - S para indicar si se debe almacenar el signo;
        - 9 (tamaño de la variable en posiciones a la izquierda del punto - unidades);
        - V para indicar el punto decimal;
        - 9 (tamaño de la variable en posiciones a la derecha del punto – decimales);
        - Ejemplos:
            - PIC 9(1)
            - PIC 9(2)
            - PIC 9(2)V9(2)
            - PIC S9(2)V9(2)
            - PICTURE 9(2)
            - PICTURE 9(2)V9(2)
            - PICTURE S9(2)V9(2)

    - Alfabéticas (solo letras):
        - A (tamaño de la variable);
        - Ejemplos:
            - PIC A(1)
            - PIC A(5)
            - PICTURE A(5)

    - Alfanuméricas (cualquier carácter):
        - X (tamaño de la variable);
        - Ejemplos:
            - PIC X(1)
            - PIC X(5)
            - PICTURE X(5)

- VALUE es una literal del tipo de datos respectivo o bien una constante figurativa (ZEROS o SPACES); es opcional. Si el tipo de datos no es numérico, la constante de texto debe ir entre comillas simples.
- NOTA1: si después de la ”especificacion-formato” o de la clásula VALUE y su valor respectivo se detecta que viene otro lexema que corresponde a una palabra reservada de COBOL (por ejemplo: JUSTIFIED) entonces se puede asumir que desde esa palabra reservada y hasta el punto final de la línea la sintaxis es correcta; esto se hace porque la idea es permitir otras cláusulas de COBOL dentro de una instrucción de KOBOL; en otras palabras, su programa solo debe validar hasta la cláusula VALUE.
- NOTA2: una línea que muestre cualquier otro número de nivel, que sea de 01 a 49 o el 66 o el 88 implica que el resto de la línea se debe ignorar y dar por correcta, sin tener que realizar ninguna verificación. Niveles o números más allá del 49 y que no sean 66, 77 u 88 no están permitidos. También, tomar en cuenta que con excepción del 01, todos los otros niveles van en el margen B.

En cuanto a la conversión de datos:

- No se permite que una variable de texto se asigne a una numérica o viceversa; es error y se debe reportar.
- Si a una variable de cierta precisión numérica se le asigna otra numérica o una expresión numérica cuyo resultado es de mayor precisión pues no es error, puesto que en tal caso el resultado o valor se convierte al tipo de datos de menor precisión y se asigna a la variable sin problemas; en este caso no debe validarse nada, basta con asegurarse que la variable o la variable/expresión que se asigna sean ambas numéricas.

Ejemplos:

```
77 NUMERO1    PIC 9(2) VALUE ZEROS.
77 NUMERO2    PIC 9(2) VALUE ZEROS.
77 RESULTADO1 PIC 9(2)V9(2) VALUE ZEROS.
77 RESULTADO2 PIC 9(2)V9(2) VALUE ZEROS.
77 RESULTADO3 PIC 9(2)V9(2) VALUE ZEROS.
77 RESULTADO4 PIC S9(2)V9(2) VALUE ZEROS.
77 RESULTADO5 PIC 9(2)V9(2) VALUE ZEROS.
       77 RESPUESTA PICTURE A(2)   VALUE "SI".
       77 DESCRIPCION PICTURE A(200) VALUE SPACES.
```

ERRORES

KOVOL debe reportar al usuario los errores que detecta cuando analiza las hileras 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 8 de la línea respectiva en el archivo de errores, de manera que así se facilite asociar, visualmente, los errores a la línea del programa Kobol 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.

Ejemplo:

```
00001  IDENTIFICATION DIVISION.
00002   PROGRAM-ID. CALCULOS.
00003   AUTHOR. MARIO QUIROS.
00004 *
00005 *<-- Columna 7 con asterisco significa comentario
00006 *
00007 * CURSO COMPILADORES                    esta es la columna 72--> *
00008 *
00009 *8901<-- Este es el margen A (columnas 8 a 11)
00010 *    2<-- Aqui empieza el margen B (columna 12)
00011 *
00012  ENVIRONMENT DIVISION.
00013
00014  DATA DIVISION.                                                EstasSon
00015   wORKING-STORAGE SECTION.                                     columnas
00016    77 NUMERO1    PIC 9(2) VALUE ZEROS.                         73 A 80
00017    77 NUMERO2    PIC 9(2) VALUE ZEROS.
00018    77 RESULTADO1 PIC 9(2)V9(2) VALUE ZEROS.
00019    77 RESULTADO2 PIC 9(2)V9(2) VALUE ZEROS.
00020    77 RESULTADO3 PIC 9(2)V9(2) VALUE ZEROS.
00021    77 RESULTADO4 PIC S9(2)V9(2) VALUE ZEROS.
00022    77 RESULTADO5 PIC 9(2)V9(2) VALUE ZEROS.
00023
00024  PROCEDURE DIVISION.
00025   INICIO.
00026 *     <-- Aqui empieza el margen B (columna 12)
00027      DISPLAY "PRIMER  NUMERO: " WITH NO ADVANCING.
00028      ACCEPT NUMERO1$.
       ERROR 20: Identificador solo debe llevar letras, números y guiones.
       ERROR 33: Variable no definida.
00029      DISPLAY "SEGUNDO NUMERO: " WITH NO ADVANCING.
00030      ACCEPT NUMERO2.
00031      COMPUTE RESULTADO1 = NUMERO1 * NUMERO2.
00032      COMPUTE RESULTADO2 = NUMERO1 / NUMERO2.
00033      COMPUTE RESULTADO3 = NUMERO1 + NUMERO2.
00034      COMPUTE RESULTADO4 = NUMERO1 - NUMERO2.
00035      COMPUTE RESULTADO5 = NUMERO1 * NUMERO1 * (NUMERO2 * NUMERO2).
00036      DISPLAY "MULTIPLICACION: ", RESULTADO1.
00037      DISPLAY "DIVISION      : ", RESULTADO2.
00038      DISPLAY "SUMA          : ", RESULTADO3.
00039      DISPLAY "RESTA         : ", RESULTADO4.
00040      DISPLAY "EXPRESION     : ", RESULTADO5.
00041      STOP RUN.
```

ASIGNACIÓN DE VARIABLES
La asignación de valores a las variables se hace mediante el signo de igual o =.

Ver el comando COMPUTE más adelante.

Se recuerda que:

- Un operador aritmético o un signo igual tiene que ir precedido por un espacio y seguido por un espacio.

COMANDOS DE KOVOL
Se describen a continuación cada uno de los comandos que maneja KOVOL.

ACCEPT
Sintaxis: **ACCEPT** Variable
Permite ingresar valores a las variables desde el teclado.
Se puede asumir que solo se pide una variable a la vez.
La variable debe haber sido definida previamente.

Ejemplos:
ACCEPT NUMERO1.
ACCEPT NUMERO2.
ACCEPT RESPUESTA.

COMPUTE
Sintaxis: COMPUTE variable ROUNDED = expresión
Para asignar el valor de una expresión a una variable.
La variable debe haber sido definida previamente.
La expresión debe ser de un tipo compatible con el de la variable.
ROUNDED es para indicar que se debe redondear el resultado; es opcional.
IMPORTANTE: Se debe validar que la expresión sea correcta y válida; la misma puede incluir: caracteres de agrupación (paréntesis), constantes, variables previamente definidas (si no lo han sido es error y se debe reportar) y operadores aritméticos (solo los que se citaron arriba).

Ejemplos:
```
COMPUTE RESULTADO5 = NUMERO1 * NUMERO1 * (NUMERO2 * NUMERO2).
```

Se recuerda que:

- Un paréntesis izquierdo no puede ir seguido por un espacio.
- Un paréntesis derecho no puede ir precedido por un espacio.
- Un operador aritmético o un signo igual tiene que ir precedido por un espacio y seguido por un espacio.

DISPLAY
Sintaxis: DISPLAY “Texto”, Variable
Para imprimir en la pantalla.
Se puede asumir que solo se imprime un texto y una variable a la vez.
El texto es opcional; debe ir delimitado por comillas dobles.
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).

Ejemplos:

```
           DISPLAY "MULTIPLICACION: ", RESULTADO1.
           DISPLAY "DIVISION      : ", RESULTADO2.
           DISPLAY "SUMA          : ", RESULTADO3.
           DISPLAY "RESTA         : ", RESULTADO4.
           DISPLAY "EXPRESION     : ", RESULTADO5.
```

STOP RUN
Sintaxis: STOP RUN
Termina la ejecución del programa.

Ejemplo:
STOP RUN.

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

Ejemplos:
DISPLAY "PRIMER NUMERO: " WITH NO ADVANCING.

Entrega del proyecto
Se deberá entregar como proyecto:

- 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.
    - El programa debe compilar sin errores.

- Un manual técnico que explique cómo instalar y ejecutar el compilador KOVOL 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, pero si para revisar el código. La mejor manera de estar seguros de que se ejecute bien, es ir a otra computadora y ejecutarlo ahí, asegurándose que no dé problemas. 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.

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. Procesamiento adecuado de la PROCEDURE DIVISION y de los nombres de párrafo programados (revisar que vengan siguiendo la normativa indicada en el enunciado)	15	8	4	0
2. Uso correcto del comando ACCEPT	20	10	5	0
3. Uso correcto del comando DISPLAY	20	10	5	0
3. Uso correcto del comando COMPUTE	40	20	10	0
3. Uso correcto del comando STOP RUN	5	2	1	0
TOTAL	**100**	**50**	**25**	**0**

ANEXO:
Lista de palabras reservadas de COBOL

A

ACCEPT
ACCESS
ADD
ADDRESS
ADVANCING
AFTER
ALL
ALPHABET
ALPHABETIC
ALPHABETIC-LOWER
ALPHABETIC-UPPER
APHANUMERIC
ALPHANUMERIC-EDITED
ALSO
ALTER
ALTERNATE
AND
ANY
APPLY
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR
AUTO
AUTO-SKIP
AUTOMATIC
AUTOTERMINATE

B

BACKGROUND
BACKGROUND-COLOR
BACKGROUND-HIGH
BACKGROUND-LOW
BACKWARD
BEEP
BEFORE
BELL
BIND
BINARY
BLANK
BLINK
BLINKING
BLOCK
BOLD
BOTTOM
BY

C

CALL
CANCEL
CD
CF
CH
CHARACTER
CHARACTERS
CLASS
CLOSE
COBOL
CODE
CODE-SET
COL
COLLATING
COLUMN
COMMA
COMMUNICATION
COMP
COMP-1
COMP-2

COMP-3
COMP-4
COMP-6
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTATIONAL-4
COMPUTE
CONFIGURATION
CONSOLE
CONTAINS
CONTENT
CONTINUE
CONTROL
CONTROLS
CONVERSION
CONVERT
CONVERTING
COPY
CORR
CORRESPONDING
COUNT
CRT
CURRENCY
CURSOR

D

DATA
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DAY-OF-WEEK
DE
DEBUG
DEBUGGING
DECIMAL-POINT
DECLARATIVES
DEFAULT
DELETE
DELIMITED

DELIMITER
DEPENDING
DESCENDING
DESTINATION
DETAIL
DISABLE
DISPLAY
DIVIDE
DIVISION
DRAWN
DUPLICATES
DYNAMIC

E

ECHO
EGI
ELSE
EMI
EMPTY-CHECK
ENABLED
END
END-ACCEPT
END-ADD
END-CALL
END-COMPUTE
END-DELETE
END-DIVIDE
END-EVALUATEQ
END-IF
END-MULTIPLY
END-OF-PAGE
END-PERFORM
END-READ
END-RECEIVE
END-RETURN
END-REWRITE
END-SEARCH
END-START
END-STRING
END-SUBTRACT
END-UNSTRING

END-WRITE
ENTER
ENVIRONMENT
EOL
EOP
EOS
EQUAL
ERASE
ERROR
ESCAPE
ESI
EVALUATE
EVERY
EXCEPTION
EXCLUSIVE
EXIT
EXTEND
EXTERNAL

F

FALSE
FD
FILE
FILE-CONTROL
FILE-ID
FILE-PREFIX
FILLER
FINAL
FIRST
FOOTING
FOR
FOREGROUND-COLOR
FOREGROUND-COLOUR
FROM
FULL

G

GENERATE
GIVING
GLOBAL
GO

GOBACK
GREATER
GRID
GROUP

H

HEADING
HIGH
HIGH-BALUE
HIGH-VALUES
HIGHLIGHT

I

I-O
I-O-CONTROL
IDENTIFICATION
IF
IN
INDEX
INDEXED
INDICATE
INITIAL
INITIALIZE
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS

J

JUST
JUSTIFIED

K

KEY

L

LABEL
LAST
LEADING
LEFT
LEFTLINE
LENGTH
LENGTH-CHECK
LESS
LIMIT
LIMITS
LINAGE
LINGAKE-COUNTER
LINES
LINKAGE
LOCK
LOCK-HOLDING
LOW
LOW-VALUE
LOW-VALUES
LOWLIGHT

M

MEMORY
MERGE
MODE
MODULES
MOVE
MULTIPLE
MULTIPLY

N

NATIVE
NEGATIVE
NEXT
NO
NO-ECHO
NOT
NUMBER

NUMERIC
NUMERIC-EDITED

O

OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
OPEN
OPTIONAL
OR
ORDER
ORGANIZATION
OTHER
OTHERS
OUTPUT
OVERFLOW
OVERLINE

P

PACKED-DECIMAL
PADDING
PGE
PAGE-COUNTER
PERFORM
PIC
PICTURE
PLUS
POINTER
POS
POSITION
POSITIVE
PREVIOUS
PRINT-CONTROL
PRINTING
PROCEDURE
PROCEDURES
PROCEED
PROGRAM

PROGRAM-ID
PROMPT
PROTECTED
PURGE

Q

QUEUE
QUOTE
QUOTES

R

RANDOM
RD
READ
READERS
RECEIVE
RECORD
RECORDING
RECORDS
REDEFINES
REEL
REFERENCE
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMOVAL
RENAMES
REPLACE
REPLACING
REQUIRED
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RETURN
RETUNRNIN
RETURN-CODE
RETURN-UNSIGNED

REVERSE
REVERSE-VIDEO
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROLLBACK
RUN

S

SAME
SCREEN
SD
SEARCH
SECTION
SECURE
SECURITY
SEGMENT
SEGMENT-LIMIT
SELECT
SEND
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SIGN
SIZE
SORT
SORT-MERGE
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
START
STATUS
STOP
STRING

SUBTRACT
SUPPRESS
SYMBOLIC
SYNC
SYCHRONIZED

T

TAB
TALLYNG
TAPE
TERMINAL
TERMINATE
TEST
TEXT
THAN
THEN
THROUGH
THRU
TIME
TIMES
TO
TOP
TRAILING
TRUE

U

UNDERLINE
UNDERLINED
UNIT
UNLOCK
UNSTRING
UNTIL
UP
UPDATE
UPDATES
UPON
USAGE
USE
USING

V

VALUE
VALUES
VARYING

W

WHEN
WITH
WORDS
WORKING-STORAGE
WRITE
WRITERS

Z

ZERO
ZEROS
ZEROES

Fin.