	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Marco Antonio Martinez Quintana
Asignatura:	Fundamentos de Programación
Grupo:	03
No de Práctica(s):	Practica N°5
Integrante(s):	Torres Gracian Christian Ivan
No. de Equipo de cómputo empleado:	No Aplica
No. de Lista o Brigada:	55
Semestre:	Semestre N°1
Fecha de entrega:	01/11/2020
Observaciones:	

CALIFICACIÓN: _____

Objetivo:
Elaborar pseudocódigos que representen soluciones algorítmicas empleando la sintaxis y semántica adecuadas.

Actividades:

- Elaborar un pseudocódigo que represente la solución algorítmica de un problema en el cual requiera el uso de la estructura de control de flujo condicional.
- A través de un pseudocódigo, representar la solución algorítmica de un problema en el cual requiera el uso de la estructura de control iterativa.

Introducción
Una vez que un problema dado ha sido analizado (se obtiene el conjunto de datos de entrada y el conjunto de datos de salida esperado) y se ha diseñado un algoritmo que lo resuelva de manera eficiente (procesamiento de datos), se debe proceder a la etapa de codificación del algoritmo.
Para que la solución de un problema (algoritmo) pueda ser codificada, se debe generar una representación del mismo. Una representación algorítmica elemental es el pseudocódigo.
Un pseudocódigo es la representación escrita de un algoritmo, es decir, muestra en forma de texto los pasos a seguir para solucionar un problema. El pseudocódigo posee una sintaxis propia para poder realizar la representación del algoritmo (solución de un problema).

Sintaxis de pseudocódigo
El lenguaje pseudocódigo tiene diversas reglas semánticas y sintácticas. A continuación, se describen las más importantes:

1. Alcance del programa: Todo pseudocódigo está limitado por las etiquetas de INICIO y FIN. Dentro de estas etiquetas se deben escribir todas las instrucciones del programa.
2. Palabras reservadas con mayúsculas: Todas las palabras propias del pseudocódigo deben de ser escritas en mayúsculas.
3. Sangría o tabulación: El pseudocódigo debe tener diversas alineaciones para que el código sea más fácil de entender y depurar.
4. Lectura / escritura: Para indicar lectura de datos se utiliza la etiqueta LEER. Para indicar escritura de datos se utiliza la etiqueta ESCRIBIR. La lectura de datos se realiza, por defecto, desde el teclado, que es la entrada estándar del sistema. La escritura de datos se realiza, por defecto, en la pantalla, que es la salida estándar del Sistema.

Ejemplo

ESCRIBIR "Ingresar la altura del polígono"
LEER altura

5. Declaración de variables: la declaración de variables la definen un identificador (nombre), seguido de dos puntos, seguido del tipo de dato, es decir:

<nombreVariable>:<tipoDeDato>
Los tipos de datos que se pueden utilizar son:
ENTERO -> valor entero positivo y/o negativo
REAL -> valor con punto flotante y signo
BOOLEANO -> valor de dos estados: verdadero o falso
CARACTER -> valor tipo carácter
CADENA -> cadena de caracteres

Ejemplo

contador: ENTERO
producto: REAL
continuar: BOOLEANO
Es posible declarar más de una variable de un mismo tipo de dato utilizando arreglos, indicando la cantidad de variables que se requieren, su sintaxis es la siguiente:
<nombreVariable>[cantidad]:<tipoDeDato>

Ejemplo
contador[5]: ENTERO // 5 variables de tipo entero
division[3]: REAL // 3 variables de tipo real
bandera[6]: BOOLEANO // 6 variables de tipo booleano

Existe un tipo de dato compuesto, es decir, que puede contener uno o más tipos de datos simples diferentes. Este tipo de dato se conoce como registro o estructura y su sintaxis es la siguiente

<nombreRegistro>:REG
<nombreVariable_1>:<tipoDeDato>
...
<nombreVariable_N>:<tipoDeDato>
FIN REG

Para crear una variable tipo registro se debe indicar el nombre del registro y el nombre de la variable. Para acceder a los datos del registro se hace uso del operador ".".

Ejemplo
domicilio:REG
calle: CADENA
número: ENTERO
ciudad: CADENA
FIN REG
usuario:REG domicilio // variable llamada usuario de tipo registro

usuario.calle := "Av. Imán"
usuario.número := 3000
usuario.ciudad := "México"

Es posible crear variables constantes con la palabra reservada CONST, la cual indica que un identificador no cambia su valor durante todo el pseudocódigo. Las constantes (por convención) se escriben con mayúsculas y se deben inicializar al momento de declararse.

Ejemplo
NUM_MAX := 1000: REAL, CONST

6. Operadores aritméticos: Se tiene la posibilidad de utilizar operadores aritméticos y lógicos:

Operadores aritméticos: suma (+), resta (-), multiplicación (*), división real (/), división entera (div), módulo (mod), exponenciación (^), asignación (:=).
Operadores lógicos: igualdad (=), y-lógica o AND (&), o-lógica u OR (|), negación o NOT (!), relaciones de orden (<, >, <=, >=) y diferente (<>).

La tabla de verdad de los operadores lógicos AND, OR y NOT se describe a continuación:

A	B	A & B	A | B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

NOTA: A y B son dos condiciones, el valor 0 indica falso y el valor 1 indica verdadero.

7. Notación de camello. Para nombrar variables y nombres de funciones se debe hacer uso de la notación de camello

En la notación de camello (llamada así porque parecen las jorobas de un camello) los nombres de cada palabra empiezan con mayúscula y el resto se escribe con minúsculas. Existen dos tipos de notaciones de camello: lower camel case que en la cual la primera letra de la variable inicia con minúscula y upper camel case en la cual todas las palabras inician con mayúscula. No se usan puntos ni guiones para separar las palabras (a excepción de las constantes que utilizan guiones bajos). Además, para saber el tipo de variable se recomienda utilizar un prefijo.

Ejemplo
// variables
realAreaDelTriangulo: REAL // lower camel case
EnteroRadioCirculo: REAL // upper camel case
// funciones
calcularArea()
obtenerPerimetro()

Estructuras de control de flujo

Las estructuras de control de flujo permiten la ejecución condicional y la repetición de un conjunto de instrucciones.
Existen 3 estructuras de control: secuencial, condicional y repetitivas o iterativas.
Estructura de control secuencial
Las estructuras de control secuenciales son las sentencias o declaraciones que se realizan una a continuación de otra en el orden en el que están escritas.

Ejemplo
INICIO
x : REAL
x := 5.8
*x := x * 2*
FIN

Estructuras de control condicionales (o selectivas)

Las estructuras de control condicionales permiten evaluar una expresión lógica (condición que puede ser verdadera o falsa) y, dependiendo del resultado, se realiza uno u otro flujo de instrucciones. Estas estructuras son mutuamente excluyentes (o se ejecuta una acción o se ejecuta la otra)
La estructura de control de flujo más simple es la estructura condicional SI, su sintaxis es la siguiente:

SI condición ENTONCES
[Acción]
FIN SI

Se evalúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque [Acción]. Si no se cumple la condición, se continúa con el flujo normal del programa.

Funciones

Cuando la solución de un problema es muy compleja se suele ocupar el diseño descendente (divide y vencerás). Este diseño implica la división de un problema en varios subprocesos más sencillos que juntos forman la solución completa. A estos subprocesos se les llaman métodos o funciones.
Una función está constituida por un identificador de función (nombre), de cero a n parámetros de entrada y un valor de retorno:

INICIO
FUNC identificador (var:TipoDato,..., var:TipoDato) RET: TipoDato
[Acciones]
FIN FUNC
FIN

El identificador es el nombre con el que llama a la función. Las funciones pueden o no recibir algún(os) parámetro(s) (tipo(s) de dato(s)) como entrada; si la función recibe alguno se debe incluir entre los paréntesis. Todas las funciones pueden regresar un valor al final de su ejecución (el resultado).
Todas las estructuras de control de flujo (secuencial, condicional y repetitivas o iterativas) deben ir dentro de alguna función.

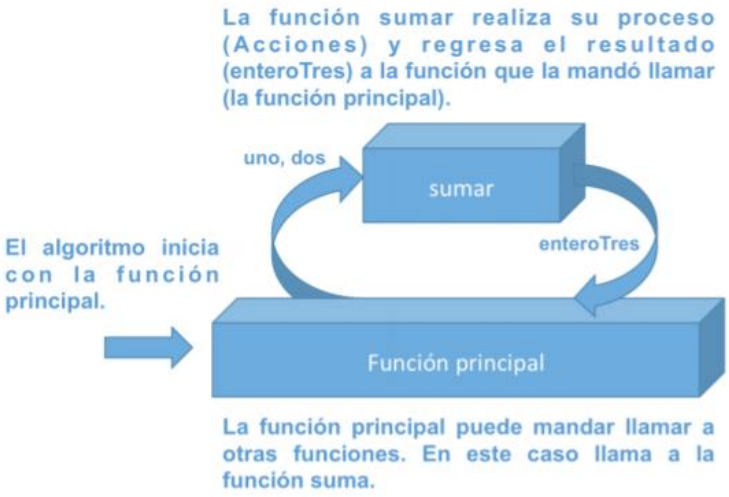
Ejemplo

```
INICIO
     FUNC principal (vacío) RET: vacío
          a, b, c: ENTERO
          a := 5
          b := 24
          c := sumar(a, b)
          ESCRIBIR c
     FIN FUNC
FIN

INICIO
     ** Función que suma dos números enteros
     FUNC sumar (uno:ENTERO, dos: ENTERO) RET: ENTERO
          enteroTres: ENTERO
          enteroTres:= uno + dos
          RET enteroTres
     FIN FUNC
FIN


// >>> 29
```

Descripción

La primera función que se ejecuta es 'principal', ahí se crean las variables (uno y dos) y, posteriormente, se manda llamar a la función 'sumar'. La función 'sumar' recibe como parámetros dos valores enteros y devuelve como resultado un valor de tipo entero, que es la suma de los valores que se enviaron como parámetro.
Para la función 'principal' los pasos que realiza la función 'sumar' son transparentes, es decir, solo manda a llamar a la función y espera el parámetro de retorno.

La siguiente figura permite analizar el pseudocódigo a través del tiempo. El algoritmo inicia con la función principal, dentro de esta función se hace una llamada a una función externa (sumar). Sumar realiza su proceso (ejecuta su algoritmo) y devuelve un valor a la función principal, la cual sigue su flujo hasta que su estructura secuencial (las instrucciones del pseudocódigo) llega a su fin.



Resultados

```
01.    ALGORITMO Sumar;
02.    VAR
03.        ENTERO Numero1, Numero2, Resultado;
04.    INICIO
05.        ESCRIBIR("Dime dos números para sumar: ");
06.        LEER(Numero1, Numero2);
07.
08.        Resultado <- Numero1 + Numero2;
09.
10.        ESCRIBIR("La suma es: ", Resultado);
11.    FIN
```

Bibliografía

- Metodología de la programación. Osvaldo Cairó, tercera edición, México D.F., Alfaomega 2005.
- Metodología de la programación a través de pseudocódigo. Miguel Ángel Rodríguez Almeida, primera edición, McGraw Hill