UNIVERSIDAD DE CARABOBO
FACULTAD EXPERIMENTAL DE CIENCIAS Y TECNOLOGÍA
DEPARTAMENTO DE COMPUTACIÓN
GRUPO DE DESARROLLO DE SOFTWARE Y SISTEMAS
CS0118 ALGORITMOS Y PROGRAMACIÓN I

NOTACIÓN ALGORITMICA

1. Estructura de un Algoritmo

La sintaxis de un algoritmo está definida por la siguiente estructura:

Nombre del Algoritmo

// Declaración de constantes

// Declaración de tipos

// Declaración de variables Globales

// Declaración de funciones y procedimientos

// Cuerpo del algoritmo
Inicio

Declaración de Variables locales
Acciones…

Fin

La sintaxis de cada una de estas secciones será desarrollada más adelante en este documento.

2.- Comentarios

Un comentario es una información al lector del programa y no realiza ninguna instrucción ejecutable, sólo tiene efecto de documentación interna del programa.

Cualquier comentario dentro del algoritmo estará precedido por " // "

Ejemplo:

// Esto es un ejemplo de un comentario

3.- Presentación

- *Palabras Reservadas* (palabras en español similares a sus homónimas utilizadas en los lenguajes de programación), deberán ser subrayadas. Ejemplo: Algoritmo, Mientras, Si, Para, etc.

- *Identación*, margen o sangría del lado izquierdo que permite la justificación del texto al escribir un algoritmo.

  Ejemplo:

  ```
  Si (Condición) Entonces
  Inicio
        Escribir(‘ Hola Mundo‘)
        Escribir(‘ Este es mi primer Programa ‘)

  Fin_si
  ```

- Se escribirá una acción por línea.

4.- Nombre del Algoritmo

Es el nombre del algoritmo que se va a desarrollar. Normalmente se utiliza un nombre nemotécnico, asociado a la función que cumplirá el algoritmo en cuestión.

Ejemplo:

Algoritmo Sumar

5.- Constantes

Cada constante tiene un identificador único. Dicho identificador es expresado utilizando únicamente letras mayúsculas. La sintaxis es la siguiente:

Const

$IDENTIFICADOR_1 \leftarrow valor_1$
$IDENTIFICADOR_2 \leftarrow valor_2$
:
$IDENTIFICADOR_n \leftarrow valor_n$

Donde:

§ $IDENTIFICADOR_i$ es el identificador (nombre) asociado a la i-ésima constante $(1 \leq i \leq n)$.
§ $valor_i$ es el valor asociado a la i – ésima constante $(1 \leq i \leq n)$.

Ejemplo:

Const

MÁXIMO ← 50

6.- Variables

Cada variable tiene un identificador único, y puede tomar ciertos valores pertenecientes a un dominio. El identificador debe ser escrito utilizando únicamente letras minúsculas. La sintaxis de declaración de variables es la siguiente:

Var

$identificador_{11}$, $identificador_{12}$, …, $identificador_{1k1}$: $\underline{T_1}$
$identificador_{21}$, $identificador_{22}$, …, $identificador_{2k2}$: $\underline{T_2}$
:
:
$identificador_{n1}$, $identificador_{n2}$, …, $identificador_{nkn}$: $\underline{T_n}$

Donde:

- $identificador_{ij}$ es el identificador (nombre) asociado a la j-ésima variable del i-ésimo tipo ($1 \leq i \leq n$).
- $\underline{T_i}$ es el identidicador (nombre) asociado al i-ésimo tipo ($1 \leq i \leq n$).

Ejemplo:

Var

contador , edad : entero
acumulador : real

7.- Tipos de datos

a.- Tipos elementales o primitivos

i. Tipo lógico

- Identificador: Lógico
- Dominio: falso, verdadero
- Operaciones: $\neg$ (negación) , $\wedge$ (conjunción) , $\vee$ (disyunción)
- Relaciones: $\equiv$ (equivalencia)

Ejemplo: p≡verdadero y q≡falso
$p \wedge q \equiv$ falso

ii. Tipo entero

- Identificador: Entero
- Dominio: Z como el conjunto de los números enteros.
  { ... -3 , -2 , -1 , 0 , 1 , 2 , 3 ... }

§ Operaciones: +, -, *, div, mod, abs
§ Relaciones: =, ≠, >,<, ≤, ≥

Ejemplo: 785
-27
5 div 3 = 1
5 mod 3 = 2

iii. **Tipo real**

§ Identificador: Real
§ Dominio: R como el conjunto de los números reales
§ Operaciones: +, -, *, /
§ Relaciones: =, ≠, >,<, ≤, ≥

Ejemplo: 78.3
-22

iv. **Tipo caracter**

§ Identificador: Caracter
§ Dominio: {'A',...,'Z'} U {'a',...,'z'} U {'0',...,'9'} U {'$','#','&',....}
§ Operaciones: funciones ordinales
§ Relaciones: =, ≠, >,<, ≤, ≥

Ejemplo: 'A' ≠ 'B'

v. **Tipo cadena**

§ Identificador : Cadena
§ Dominio: Secuencias de caracteres
§ Operaciones: primero (primer carácter de una cadena), resto (elimina el primer carácter y retorna el resto de la cadena), último (último carácter de una cadena),& (concatenación), longitud (número de caracteres de una cadena)
§ Relaciones: =, ≠, >,<, ≤, ≥

Ejemplo: 'Maria' = 'Dayana' es igual a falso
Primero ('ferrocarril') retorna 'f'

b.- Tipos compuestos o estructurados

Un tipo compuesto está conformado por otros tipos de datos (primitivos o estructurados) definidos previamente. Estos tipos deben declararse. Una declaración de un tipo compuesto especifica el identificador del tipo y su definición. La sintaxis para la declaración de un tipo compuesto es la siguiente:

Tipo

$tipo_1$ = <def_1 >
$tipo_2$ = <def_2 >
:
:
$tipo_m$ = <def_m >

Donde:
§ $tipo_i$ es el identificador del i-ésimo tipo compuesto ($1 \leq i \leq m$)
§ def_m es la definición del i-ésimo tipo compuesto ($1 \leq i \leq m$)

Ejemplo :

Tipo

Numero = Entero

Var

contador : Numero

i. Tipos Ordinales

Un tipo *T* es ordinal si y solo sí sus elementos están organizados de manera tal que satisfacen las siguientes propiedades:

- Existen dos elementos notables en *T*: un primer elemento y un último elemento.
- Para cada elemento en *T*, excepto el último existe un único elemento llamado sucesor o siguiente.
- Para cada elemento en *T*, excepto el primero existe un único elemento que le precede llamado predecesor o anterior.
- Cada elemento de *T* tiene un valor entero asociado a él, llamado número ordinal o índice, que indica la posición del elemento en la secuencia de valores que define a *T*.

Observación: Todos los tipos de datos primitivos, con excepción de los tipos real y cadena, son tipos ordinales, porque no satisface las tres últimas propiedades de la definición de tipo ordinal.

Definición 2 (Funciones ordinales)
Sea *T* un tipo de dato ordinal. Se definen entonces las funciones ordinales como sigue:

§ indice : T → Z; retorna el número ordinal de su argumento.
§ anterior : T → T; retorna el predecesor de su argumento.
§ siguiente : T → T; retorna el sucesor de su argumento.

ii. Tipo Enumerado

Es una secuencia ordenada y finita de valores referenciados por identificadores. La sintaxis de un enumerado es la siguiente:

Tipo

$T = (ident_1, ident_2, ident_3, ident_4, ..., ident_n)$

Donde :

- T es el identificador del nuevo tipo.
- Los $ident_i$ $(1 \leq i \leq n)$ son los identificadores que forman el tipo.
- El orden de los identificadores en T está definido por la siguiente regla:

$$\forall i \forall j : (1 \leq i, j \leq n) : [(ident_i < ident_j) \Leftrightarrow (i < j)]$$

Características:

- Una variable *x* de tipo Enumerado *T* sólo puede tomar valores que se encuentren dentro de la lista de identificadores que define a *T*.

- Es importante acotar que un identificador no puede pertenecer a más de un tipo enumerado.

- Los únicos operadores asociados al tipo Enumerado son el operador de asignación y los operadores de comparación.

Ejemplo :

Tipo

Meses = (‘enero’ , ‘febrero’, . . . , ‘diciembre’)

iii. Tipo Intervalo

La sintaxis de un intervalo es la siguiente:

Tipo:

T = min ... max

Donde:
- T es el identificador del nuevo tipo.
- min y max especifican los límites inferior y superior del intervalo.
- El tipo de datos de min y max puede ser entero o carácter o un tipo Enumerado.
- Se puede verificar que la definición del intervalo es aceptable si y solo si se verifica que min $\leq$ max.

Las operaciones que se pueden realizar sobre una variable de tipo Intervalo son iguales que las del tipo a partir del cual está definido.

Ejemplo :

Tipo

Nota = 0 ... 20

iv. Tipo Arreglo

La sintaxis a seguir para definir un tipo arreglo de r dimensiones ($r \geq 1$), tiene la siguiente estructura:

Tipo

T = arreglo [$m_{1...}n_1$, $m_{2...}n_2$, ..., $m_{r\ ...}\ n_r$] de T_0

Donde :

- T es el identificador del nuevo tipo.. Los intervalos $m_{1...}n_1$, $m_{2...}n_2$, ..., $m_{r...}n_r$ representan los índices del arreglo para cada una de sus dimensiones. El tipo del índice debe ser un tipo ordinal
- T_0 es el tipo base del arreglo y puede ser primitivo o estructurado

Ejemplo :

Tipo

Nombres = arreglo [1 .. 10] de cadena

v. Tipo Registro

Los componentes de un registro se conocen como campos. Cada campo posee un identificador único y pertenece a un tipo previamente definido. La sintaxis para definir un registro es la siguiente:

Tipo
T = registro
inicio
$identificador_{11}$, $identificador_{12}$, …, $identificador_{1k1}$: T_1
$identificador_{21}$, $identificador_{22}$, …, $identificador_{2k2}$: T_2
:
:
$identificador_{n1}$, $identificador_{n2}$, …, $identificador_{nkn}$: T_n
fin_registro

Donde:

- T es el identificador del nuevo tipo.
- $ident_j$ es el identificador (nombre) del *j*-ésimo campo del *i*-ésimo tipo $(1 \leq i \leq n)$
  T_i es el tipo del *i*-ésimo campo $(1 \leq i \leq n)$

Ejemplo :

Tipo

Empleado = registro
Inicio
Nombre : cadena
Edad : entero
Dirección : cadena
Telefono : cadena
Fin_registro

vi. Tipo Apuntador

La sintaxis para la definición de un apuntador es la siguiente:

Tipo:

Apuntador = apuntador a T

Donde:

- *Apuntador* es el identificador del nuevo tipo
- *T* representa el tipo referenciado por el apuntador, este puede ser primitivo o estructurado.

8.- Acciones

8.1.- Asignación

La sintaxis para realizar una asignación simple es la siguiente:

$$Var_1 \leftarrow expr_1$$

Donde Var_1 representa la variable a la cual se le esta asignado la expresión $expr_1$.

Ejemplo: contador $\leftarrow 3 + 4$

8.2.- Condicional

La sintaxis de una acción condicional tiene la siguiente estructura:

Si <condición> entonces
    acciones…
sino
    acciones…

finsi

La <condición> retorna un valor lógico (verdadero, falso).

Ejemplo:

Si (A = B) Entonces

    C ← A + B
sino
    C ← A - B
finsi

8.3.- Selección

La sintaxis de una acción de selección tiene la siguiente estructura:

seleccion (var)
    caso val_1 : S_1
    caso val_2 : S_2
    ...
    caso val_r : S_r
    sino : S
fin_seleccion

Donde:

- val_i $(1 \leq i \leq r)$ es un valor constante de tipo ordinal que se denominan etiquetas de la sentencia
- S_i con $(1 \leq i \leq r)$ es el conjunto de acciones a ejecutar en caso de que var = val_i
- Si $\forall_i : 1 \leq i \leq r : var \neq val_i$ entonces se ejecutará el conjunto de acciones S correspondientes a la cláusula sino

Ejemplo:

Selección (a)
    caso 4 : Escribir (" El valor Ingresado es: 4 ");
    caso 5 : Escribir (" El valor Ingresado es: 5 ");
    sino : Escribir (" El valor Ingresado es incorrecto ");
fin_seleccion

8.4.- Iteración

8.4.1 Mientras

La sintaxis de una iteración *mientras* es la siguiente:

Mientras <condición> hacer
acciones…
fin_mientras

La <condición> retorna un valor lógico (verdadero, falso), la cual mientras sea verdadero ejecutará las acciones que se encuentren dentro del ciclo.

8.4.2 Para

La sintaxis de una iteración *para* es la siguiente:

Para i ← n [Incrementando / Decrementando] Hasta m hacer
acciones…
finpara

- i representa una variable tipo entero, que controla el número de iteraciones del para.
- n indica el valor inicial de i
- m representa el límite del valor de la variable i
- En caso de que n > m se usa la palabra opcional decrementando para realizar el conteo de forma decreciente.

8.4.3 Repetir

La sintaxis de una iteración *repetir* es la siguiente:

Repetir
acciones…

hasta <condición>

8.4.5 Entrada y Salida

a.- Entrada

La sintaxis de una acción de entrada tiene la siguiente estructura:

Leer (x_1, x_2, x_3,.... ,x_n)
Ejemplo: Leer (numero)

En donde x_i representa la variable que se desea leer.

b.- Salida

La sintaxis de una acción de salida tiene la siguiente estructura:

Escribir (x_1, x_2, x_3,.... ,x_n)
Ejemplo: Escribir (" El valor Ingresado es: ", numero)

En donde x_i representa la variable que se desea escribir.

9 Funciones y Procedimientos

9.1 Declaración de Funciones

La sintaxis para declarar una función es la siguiente:

Func <identificador> ($parm_1$, $parm_2$: T_1; $parm_3$, ..., $parm_n$:T_i) : T_k
Inicio
cuerpo de la función

<identificador> ← valor // Retornando el resultado de la función
fin_func

Donde:

- <identificador> es el identificador de la función.
- ($parm_1$, $parm_2$: T_1; $parm_3$, ..., $parm_n$:T_2) representan los parámetros de la función. T_i es el tipo de dato del parámetro.
- T_k representa el tipo de datos del resultado que genera la función.
- El cuerpo de la función es el conjunto de acciones que definen el cuerpo de la función

9.2 Procedimientos

La sintaxis a seguir para declarar un procedimiento es la siguiente:

Proc <identificador> ($parm_1$, $parm_2$: T_1; Ref $parm_3$, ..., $parm_n$:T_k)
Inicio
Cuerpo del procedimiento
Fin_proc

Donde:

- <identificador> es el identificador del procedimiento.
- ($parm_1$, $parm_2$: T_1; $parm_3$, ..., $parm_n$:T_k) son los parámetro del procedimiento y estos tienen alguna de estas 2 formas:
  - $parm_1$ y $parm_2$, son parámetros por valor.
  - Ref $parm_3$, ..., $parm_n$, son parámetros por referencia.

Nota: es de hacer notar que los parámetros por valor deben estar declarados antes que los parámetros por referencia y a su vez, estos últimos deben estar precedidos por la palabra reservada Ref

§ El cuerpo del procedimiento es el conjunto de acciones que definen el cuerpo del procedimiento