TEMA 2: ACCIONES ELEMENTALES

Declaraciones de variables, constantes y tipos. Instrucción de Asignación. Valor izquierdo y derecho de una variable. Acciones predefinidas. Operación de Lectura. Operación de Escritura.

NOTACIÓN ALGORÍTMICA:

A fin de garantizar la correcta interpretación del lector de los algoritmos que escribimos, debemos especificar una notación, la cual debe ser:

- Fácil de entender.
- Debe reflejar el análisis que produjo la solución al problema, de manera directa.
- Debe ser poderosa, es decir, que con pocos elementos se puedan expresar muchas cosas.

Los algoritmos manejan valores y datos para procesarlos y obtener resultados. Los conjuntos de valores que puede tomar la información pueden dividirse en dos categorías:

Valores no estructurados o elementales: son aquellos valores que no pueden dividirse en componentes. Ejemplo: la CI, la edad, un real, un entero.

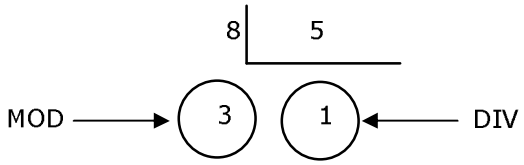
Valores estructurados: son aquellos valores que están formados por componentes y por tanto, pueden descomponerse en términos de los valores que lo constituyen. Ejemplo: una persona, un carro.

Los valores que constituyen cada conjunto se pueden expresar de tres maneras:

Constantes: denotan un valor particular dentro de un conjunto. Ejemplo: (3, 5, -10, 0) dentro del conjunto de los enteros. (verdadero, falso) lógicos. (3.5, 3.14, -10.0) reales, (‘a’, ‘b’, ‘c’) caracteres.

Variables: denotan un valor genérico dentro del conjunto. Ej A, Int, salir

Expresiones: denotan valores a partir de operaciones entre nombres y/o constantes. Ej: x+1, x^2, x>5 Y y<3, A div I, A mod I



Sobre cada uno de dichos conjuntos de valores están definidas una serie de operaciones primitivas que permiten manipular sus valores. De aquí se origina el concepto de tipo de dato.

Tipo de Dato: es un conjunto de valores junto a una serie de operaciones que pueden ser aplicadas sobre dichos valores.

Por ejemplo, objetos de tipo Entero pueden tomar valores del conjunto de los enteros (Z), y las operaciones sobre dichos valores son:
Aritméticos: +,-, -(unario), *, div, mod, ^
Relacionales: >,<, =, ≥, ≤, ≠

CARACTERÍSTICAS Y VENTAJAS DE LOS TIPOS DE DATOS

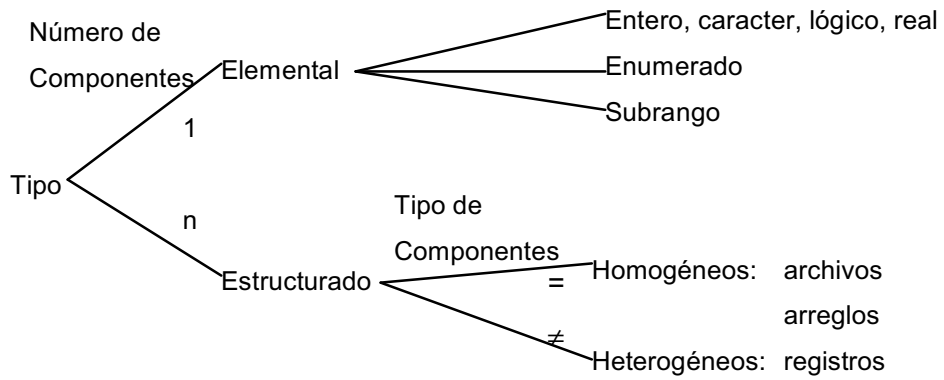
Características:
Un tipo de dato determina la clase de valores que puede asumir una variable o una expresión. En ocasiones una variable determina el tipo de una expresión.
Cada variable pertenece a un solo tipo de dato
Cada operador actúa sobre operandos de algún tipo, y da como resultado un valor de algún tipo

Ventajas:
La verificación de tipos es realizada por el traductor
Permite agrupar un conjunto de valores que guardan una estrecha relación.
Permite ver las operaciones sobre los valores de manera indivisible
Ocultamiento de la representación interna: se ve sólo la especificación y no la implementación.

CLASIFICACIÓN DE LOS TIPOS DE DATOS

El criterio para clasificar los tipos de datos está dado por el número de componentes (determina si es estructurado o no) y según el tipo de sus componentes (si es no

estructurado equivale al tipo, y si es estructurado determina si es homogéneo o heterogéneo).



Podemos referirnos a objetos de tipo no estructurado cuando el conjunto de valores que pueden tomar son no estructurados, y objetos de tipo estructurado cuando el conjunto de valores que pueden tomar son estructurados.

TIPOS DE DATOS NO ESTRUCTURADOS

Los tipos de datos no estructurados que están disponibles en la mayoría de los lenguajes de programación son el tipo entero, caracter, real y lógico. Muchos lenguajes permiten la definición de tipos subrango o subintervalos y enumerados. A continuación explicaremos brevemente cada uno de estos tipos.

Tipo Entero:

Conjunto de valores: subconjunto de los enteros

Conjunto de operaciones

Aritméticos: +,-, -(unario), *, div, mod, ^

Relacionales: >,<, =, ≥, ≤, ≠

Tipo Real:

Conjunto de valores: subconjunto de los reales

Conjunto de operaciones:

Aritméticos: +,-, -(unario), *, /, ^

Relacionales: >,<, =, ≥, ≤, ≠

Tipo Caracter:

Conjunto de valores: Tabla ASCII tiene 128 caracteres, Tabla ASCII extendida tiene 256 caracteres.

Conjunto de operaciones:

Relacionales: >,<, =, ≥, ≤, ≠

Acerca del código ASCII: La organización Internacional de Normas (ISO) define el código estadounidense ASCII (Código Standard Americano para el Intercambio de Información), que constituye el código más aceptado. Este código consta de 128 caracteres, de los cuales 99 son imprimibles y los 33 restantes son de control. En muchas máquinas se usan 8 bits para representar los caracteres, y suele usarse la tabla ASCII extendida, que consta de 256 caracteres (los mismos 128 del código ASCII y otros 128 caracteres).

En el conjunto de caracteres se tienen las 26 letras mayúsculas latinas, las 26 minúsculas, los 10 números arábigos y otros caracteres como los signos de puntuación y el espacio. Entre los caracteres hay un orden, que viene dado por el valor binario que tiene asociado. Este orden depende del sistema de codificación que se use. Sin embargo, en general los subconjuntos de letras y números son ordenados y contiguos, cumpliéndose:

‘a’<’b’< … <’z’

‘A’<’B’< … <’Z’

‘0’<’1’< … <’9’

Esto hace que los operadores relacionales puedan aplicarse sobre los caracteres. Así mismo, muchos lenguajes de alto nivel proveen herramientas para realizar operaciones aritméticas sobre caracteres. En Pascal Succ(‘a’)=’b’; Chr(ORD(‘a’)+1)=’b’. En C y en Java, sencillamente ‘a’+1=’b’. Las letras A…Z están ubicadas del 65 al 90, de la a … z del 97 al 122. Los números de 0 … 9 del 49 al 57, el espacio en el 32.

Tipo Lógico:

Conjunto de valores: {verdadero y falso}

Conjunto de operaciones:

Lógicas: no, o, y

Relacionales: =, ≠

Para recordar…

p	q	¬p	¬q	p∧q	p∨q
V	V	F	F	V	V
V	F	F	V	F	V
F	V	V	F	F	V
F	F	V	V	F	F

Resumiendo:

Tipo de dato	Constantes	Variables	Expresiones
Entero	3, 0, -1	I, J, K, ini	I+J*K, 5+8*3
Real	3.141, 0.1, -1.1	Pi, X, Z	X^(1/2) , Pi*R^2
Lógico	verdadero/falso	L1, L2, L3	no L1 y L2 o no L3
Caracter	'A', 'h', ' '	C1, C2	'A'>'B', C1≠C2

Ejercicios Resueltos:

Traducir a lenguaje algorítmico

1) ¬p∨q

```
no p o q;
```

2) ax^2+bx+c

```
a * (x^2) + b * x + c;
```

3) $\frac{a+2b}{3}$

```
(a + 2 * b) / 3;
```

PRIORIDAD DE OPERADORES

Cuando mezclamos en una expresión operadores relacionales, booleanos y aritméticos, tendremos la siguiente prioridad:

( )

- (unario), no, ^

*, /, mod, div

+, -

<, ≤ , >, ≥

=, ≠

y

o

TEMA 3: ACCIONES ELEMENTALES

Declaraciones de variables, constantes y tipos. Instrucción de Asignación. Valor izquierdo y derecho de una variable. Acciones predefinidas. Operación de Lectura. Operación de Escritura.

DECLARACIÓN DE VARIABLES

Las variables se declaran de la siguiente forma:

```
<nombre del tipo> <lista de 1 o más variables>;
```

Cada variable es la concatenación de uno o más caracteres alfanuméricos, comenzando *siempre* por un caracter alfabético.

Ejemplos:

```
Acción Principal
      Entero i, j, k;
      Real x, y;
Lógico b1;
Caracter y1;
facción
```

ACCIONES ELEMENTALES

Las acciones elementales son: la asignación, la entrada simple y la salida simple. Con estas acciones se pueden crear acciones más complejas mediante las formas de composición de acciones.

Asignación:

Asociación de un valor a una variable. Permite modificar el estado de los objetos del algoritmo. La notación algorítmica que usaremos para la asignación es:

```
<nombre de variable> ←—— <constante, variable o expresión>;
```

Por ejemplo, si I y J son variables de tipo entero, las siguientes asignaciones son validas:

```
I ← 1;            # asignación de una constante a una variable
I ← J;            # asignación de una variable a otra
I ← I+1;          # asignación del resultado de evaluar una expresión a
# una variable
I ← J mod 2;      # asignación del resultado de evaluar una expresión a
# una variable
```

Otro ejemplo

```
Caracter C1;
Entero I;
Lógico Enc;

C1 ← 'a';          # en este momento C1='a'
Enc ← (1>0);       # Luego de la asignación, el valor de Enc es
# verdadero
I ← 5+16 div 3     # luego de la asignación, el valor de I es 10
```

Secuenciamiento:

Consiste en presentar las acciones en un orden. Las acciones se efectúan en el mismo orden en que son escritas. El símbolo punto y coma (;) se utiliza para separar una instrucción de otra.

Ejemplo:

```
Acción Principal
Entero I;
      I ← 1;            #en este momento I tiene el valor de 1
      I ← I+1;          # en este momento I tiene el valor de 2
      I ← I^5;          # en este momento I tiene el valor de 32
I  ← I mod 2;      # I = 0
facción
```

Otro ejemplo

```
Acción Principal
Caracter C1;
Entero I;
Lógico Enc;
C1 ← 'a';          # en este momento C1='a'
Enc ← (1>0);       # Luego de la asignación, el valor de Enc es
# verdadero
I ← 5+16 div 3     # luego de la asignación, el valor de I es 10
facción
```

Entrada Simple

Almacena un valor extraído desde un dispositivo externo (del cual hacemos abstracción, aunque generalmente es el teclado) en una variable. El efecto de esta acción es el cambio de valor de la variable en cuestión.
La sintaxis algorítmica que adoptaremos para esta acción es:

```
Leer (<nombre de variable>);
```

Ejemplos:

```
Entero I;
Leer (I);

Real X;
Leer (X);

Caracter c;
Leer (c);
```

Cabe destacar que luego de leer un valor de un dispositivo externo, el valor de la variable cambia de igual forma que en una asignación.

Salida Simple

Transmite un valor (constante, variable o evaluación de una expresión) a un dispositivo externo (del cual hacemos abstracción, aunque generalmente es el monitor) para ser mostrado al usuario. Generalmente es utilizado para mostrar los resultados obtenidos.

La sintaxis algorítmica que utilizaremos para esta acción es:

```
Escribir(<Nombre de variable, constante o expresión>);
```

Ejemplos

```
Escribir (1);      # se ha escrito el número 1
Entero I;
I ← 1;
Escribir (I+1);    #se ha escrito el número 2
```

Ejercicios Resueltos:

Haga un algoritmo que dados dos números enteros devuelva el resultado de la suma de ellos.

```
Acción Principal
Entero A, B, Res;
Leer(A); Leer(B);
Res ← A + B;
Escribir(Res);
faccion
```

Dado un número entero N obtener el primer dígito (asociado a las unidades)

```
Acción Principal
Entero N, Res;          #se han declarado dos variables, con valor
# indefinido
Leer(N);                # N contiene el numero introducido por el
# usuario
Res ← n mod 10;   # Res tiene el resultado, el primer dígito de N
Escribir(Res);          # Se escribió el resultado.
faccion
```

Dado un numero entero N obtener el dígito asociado a las decenas

```
Acción Principal
Entero N, Res;          # se han declarado dos variables, con valor
# indefinido
Leer(N);                # N contiene el numero introducido por el
# usuario
Res ← N div 10;   # Res contiene el numero N sin el primer dígito
Res ← Res mod 10; # Res tiene el segundo dígito de N
Escribir(Res);          # Se escribió el resultado.
faccion
```

En este algoritmo podemos suprimir la variable Res, ya que la escritura soporta la evaluación de expresiones, finalmente el algoritmo puede ser escrito como:

```
Acción Principal
Entero N;                    # se declaro la variable N
Leer(N);                     # N contiene el numero introducido
# por el usuario
Escribir(N div 10 mod 10);    # Se escribió el resultado.
faccion
```

Dado un número entero N obtener el dígito I-esimo

En este caso el usuario debe introducir además del número, la posición del dígito a extraer. Haciendo analogía con los ejercicios anteriores, debemos extraer los I-1

primeros dígitos del numero N, como I es entero, debemos asumir, por ahora, que es mayor que 0

```
Acción Principal
Entero N, I, Res;
Leer (N);
Leer (I);
Res ← N div 10^(I-1);
Res ← Res mod 10;
Escribir (Res);
faccion
```

Dado un círculo centrado en el origen, determine si un punto pertenece al mismo.

Análisis: para empezar hay que saber cuales don los elementos que intervienen en este problema. Por un lado, tenemos el círculo. Un circulo centrado en el origen está definido como todos los puntos (x,y) tal que:

$$x^2+y^2 \leq r^2$$

Donde r es el radio del circulo y x e y las coordenadas de un punto en el plano R2. Los valores que necesitamos conocer entonces son el radio y el punto (x,y) que queremos verificar. El resultado es simplemente evaluar si el punto está dentro o no del círculo con la inecuación.

```
Acción Principal
Real r, x, y;
Lógico Res
Leer( r ); Leer (x); Leer (y);
Res ← (x*x + y^2 ≤ r*r)
Escribir (Res);
faccion
```

Nótese que podemos eliminar la variable Res si escribimos Escribir(x*x + y^2 ≤ r*r)

Dado un número entero de 3 dígitos, verificar si es palindrome.

```
Acción Principal
Entero N, D1, D3;
Lógico L;
Leer(N);
```

```
D3 ← N div 100;          // D1 tiene el valor del primer dígito  del
                         // número
D1 ← N mod 10;           // D3 tiene el valor del tercer dígito
L ← D1 = D3;             // si D1=D3 L tiene verdadero, sino tiene falso
Escribir(L);
faccion
```