

Unidad 2

ALGORITMOS Y ESTRUCTURAS DE DATOS

Objetivos

- Conocer las herramientas básicas para construcción de algoritmos (Objetivo clave de la unidad).
- Conocer los elementos de partida: Datos y procesos.
- Aprender que son Identificadores, variables, constantes.
- Reconocer los diferentes tipos de datos.
- Identificar y usar los elementos básicos de un programa.
- Conocer distintas formas de expresar algoritmos.
- Familiarizarse con el uso del pseudocódigo.

Contenidos

- Los Datos y su representación.
- Identificadores.
- Operadores.
- Variables.
 - Declaración
 - Tipos de Datos
 - Inicialización
- Constantes.
 - Literales
 - Simbólicas
 - Tipos de datos
- Expresiones.
- Algoritmos.
 - Diseño de algoritmos
 - Algoritmo, instrucción y programa.
 - Características de los algoritmos
 - Estructuras de programación
 - Formas de expresar algoritmos: pseudocódigo
 - Estructura general de un algoritmo escrito en pseudocódigo
- Instrucciones de entrada.
- Instrucciones de salida.

Introducción

- Todo algoritmo consta de:
 - **Datos**: ¿Cómo circulan (**Bus de datos, Bus de direcciones, Bus de control**) y se representan internamente (**memoria**)?.
 - **Procesos** : ¿cómo se pueden procesar esos datos?.
 - Los datos se guardan en “cajitas (**memoria**)” con nombres
 - **Identificadores**, dan nombre a
 - **Variables, Constantes**, ...que se relacionan mediante
 - **Operadores**, constituyendo
 - **Expresiones**, que forman parte de las **órdenes** definidas mediante
 - **Estructuras de control de flujo**
 - La comunicación con el exterior se establece con una **Interfaz básica**
 - **Instrucciones de entrada**. Meter datos
 - **Instrucciones de salida**. Sacar resultados.

Estructura interna de la memoria

- La memoria puede verse como un plano dividido en filas y en columnas.
- Cada fila se diferencia de otra por la situación que ocupa dentro del plano, su dirección.
- En cada fila se guardan los datos, el contenido.

Dirección

Contenido

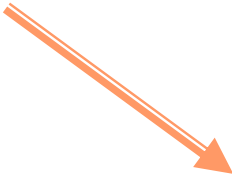
0	0	0	1	1	0	1	1	1
1	1	0	1	0	0	0	0	1
2	0	0	0	0	1	0	0	0
n	1	1	1	0	1	0	1	0

Representación : Conceptos básicos

- **Rango de representación:** Intervalo de valores que puede tomar un tipo de datos.
- **Desbordamiento de rango:** salida del intervalo.
- **La representación de un número depende de la cantidad de dígitos que se dispongan para ello.**
 - Ejem 11011 en 8 dígitos 00011011
en 16 dígitos 00000000 00011011
En 4 1011--- desbordamiento

Los Datos y su Representación

- **Dato**: Símbolo que sirve para registrar hechos, transacciones, acontecimientos,...Ej. 18, Carmen.
- **Información**: supone procesamiento de los datos para que sean útiles y significativos para quien los recibe. Ej. 18 años, 18 euros, Ópera Carmen, mi vecina Carmen.
- **Digitalización**: transformación de datos en secuencias de 0 y 1.
- **Tipo de datos**: Forma general que toma un conjunto de datos con similares características.
 - El tipo de dato determina su representación interna y las operaciones que podrán realizarse sobre él.
 - **Datos numéricos**
 - Enteros
 - Reales
 - **Datos lógicos**
 - **Datos alfanuméricos**
 - Carácter
 - Cadenas



***Ej. 18 años (entero),
18,0 euros (real),
casa nº 18 (cadena)***

Datos numéricos

Enteros (16 bits)		RANGO	Ejemplos con 8 bits		
Binario puro	$0 < \text{valor} < 2^n - 1$	65.535 positivos	$40_{(10)} = 00101000_{(2)}$		
Mód. y sig.	$- 2^{n-1} + 1 < \text{valor} < 2^{n-1} - 1$	$- 32.767 < \text{valor} < + 32.767$		Signo	Módulo
			+ 40	0	0101000
			- 40	1	0101000
			224	sale de rango	
Comp. a 1	$- 2^{n-1} + 1 < \text{valor} < 2^{n-1} - 1$	$- 32.767 < \text{valor} < + 32.767$		Signo	Módulo
			+ 40	0	0101000
			- 40	1	1010111
			224	sale de rango	
Comp. a 2	$- 2^{n-1} + 1 < \text{valor} < 2^{n-1} - 1$	$- 32.767 < \text{valor} < + 32.767$		Signo	Módulo
			+ 40	0	0101000
			- 40	1	1011000
			224	sale de rango	
Reales					
Simple precisión	31 (bit signo mantisa) 30....23 (exponente) 22...0 (mantisa) - Signo de mantisa lógica positiva Mantisa en formato exponencial Exponente en complemento a dos				
Doble precisión	63 (bit signo mantisa) 62....(exponente) 22...0 (mantisa)				

Datos lógicos o booleanos

- Se forman a partir de operaciones relacionales y lógicas, por lo que **no pueden ser datos de entrada, por tanto, NO SE LEEN.**
- Sólo toman dos valores ***verdadero*** o ***falso***.

**"NO SE LEEN" SIGNIFICA
QUE NO PUEDO METER SU
VALOR DESDE EL TECLADO**

Datos alfanuméricos

- Compuestos por caracteres alfanuméricos:
 - **Dígitos Alfabéticos**: a...z, A ...Z
 - **Dígitos Numéricos**: 0... 9
 - **Especiales**: “ - , y caracteres de control: *esc, tab,....*
- Pueden ser de dos clases
 - **Carácter**: ‘A’, ‘4’
 - **Cadena**: Teléfono: “+34954256801”
- Métodos de representación
 - **EBCDIC** (**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode)
 - **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)
 - **UNICODE**

IDENTIFICADORES

- Son nombres para referenciar objetos o procesos en los algoritmos.
- Ver pag. 6 reglas para construir identificadores
- Todos deben ser declarados antes de ser usados.
- Ej. nombreAlumno, precio, notas, IVA, numero1,....
- Algunos están definidos para realizar tareas específicas en los algoritmos: *Leer, Escribir, Mientras,....*



Palabras reservadas

OPERADORES

- **Expresión**: es un conjunto de elementos relacionados de una forma determinada.
- Los **operadores** son símbolos utilizados para representar los enlaces existentes entre los distintos elementos de una expresión.
- Se usan en programación para construir expresiones válidas.
- **Tipos**: aritméticos, relacionales, lógicos o booleanos, asignación, precedencia.

Operadores aritméticos

binarios

SÍMBOLO	OPERACIÓN
+	Suma
—	Resta
*	Multiplicación
/	División real
DIV	División entera
MOD	Resto de la división entera
↑ ^ **	Potencia (los tres símbolos suelen usarse)

Ver Ejem. pag.7

unarios

SÍMBOLO	OPERACIÓN
—	Cambio de signo

Relacionales

SÍMBOLO	OPERACIÓN
<	Menor que
==	Igual que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
!=	Distinto a

Ver Ejem. pag.8

Ej. "hoja" >= "hojas" es Falso ("hoja" **NO** es posterior o igual a "hojas")
5 < > 5 es Falso

Lógicos

SÍMBOLO	OPERACIÓN
O	Suma lógica (OR)
Y	Producto lógico (AND)
NO	Negación (NOT)

Ej. NO estoy leyendo es Falso

Ver Tablas pag. 9

Asignación y precedencia

SÍMBOLO	OPERACIÓN
= ←	ASIGNACIÓN
()	PRECEDENCIA

Ej. Edad =35 Numero ← 16.56 LETRA = 'A'

Orden de evaluación de los operadores

Paréntesis, comenzando siempre por los más internos.

Potencia.

Producto y división.

Suma y resta.

Operadores relacionales.

Operadores lógicos.

En pseudocódigo ante igual precedencia, se evalúan de izquierda a derecha.
(No tiene por qué ser así en los L. de compilación)

10 > (4 * 2)
(10 > (4 * 2)) O ('a' == 'A')
((5+2.5)*(3-5))*(-1)+5

Verdadero

Verdadero

Resultado 20

VARIABLES

- Definición: **es (el contenido de) una posición de memoria**, referenciada con un identificador, conocido como **nombre de la variable**, donde se almacena el **valor de un dato que puede variar** durante la ejecución del programa.
- Declaración: deben ser declaradas antes de ser usadas, según el siguiente formato:

Tipo nombreVariable

TIPOS de datos de las VARIABLES:

Numéricas: entero y real.

Alfanuméricas: carácter y cadena.

Booleanas: booleano.

INICIALIZACIÓN DE VARIABLES:

- Al declararlas:

Tipo nombreVariable = valorInicial

- En cualquier línea en el programa una vez declaradas

nombreVariable = valorInicial

Ejemplos:

Entero edad = 35, numero

Real precio = 0.0, IVA = 11.5

carácter estadoCivil = ' S ', letraNIF

cadena nombreAlumno.

booleano encontrado = Falso

numero = 0

nombreAlumno = " "

CONSTANTES

- **Definición:** Nombre de una posición de memoria donde se almacena el valor de un dato que no puede cambiar durante la ejecución del programa.
- Formas en que pueden aparecer en un programa:
 - **Literales:** $\text{area} = 3.14 * (\text{radio} \wedge 2)$
 - **Simbólicas:** se declaran antes de usarlas asignándoles su valor, según el siguiente formato.

NombreConstante = ValorLiteralAsociado

Ej. **PI** = 3.14

A partir de la declaración puede usarse el símbolo

$\text{area} = \mathbf{PI} * (\text{radio} \wedge 2)$

- Tipos de datos de las constantes: vamos a asociarlos al tipo de dato que almacenan, pero no es así para todos los lenguajes.

EXPRESIONES

- Una expresión está formada por constantes, variables, operadores, palabras reservadas o un conjunto de ellos, escrita siguiendo la sintaxis propia de un lenguaje.
- Los operadores permitidos en una expresión dependen de los tipos de datos de los objetos que forman la expresión.
 - Ejemplo

$$\text{area} = \mathbf{PI} * (\text{radio} \wedge 2)$$

Ver Ejem. pag.16

ALGORITMOS

- Pueden usarse distintas filosofías de diseño:
 - **Top-Down “Divide y venceras”**. Dividir el problema, programa principal, en otros más pequeños llamados módulos o subprogramas.
 - **Bottom-Up**. Identificar la estructura de datos del programa, sus funcionalidades y las interacciones entre ellos.

Para describir estas funcionalidades también se usa la técnica del Top-Down.

ALGORITMOS

■ Constan principalmente de

1. Descripción de datos: constantes, variables y **otros objetos**

2. Descripción de instrucciones

■ Características de los algoritmos:

- ❑ Eficientes
- ❑ Legibles
- ❑ Modificables
- ❑ **Modulares**
- ❑ Único punto de entrada,
único punto de salida

Constantes:

....

Variables:

....

Otros objetos:

....

Instrucción 1

Instrucción 2

Instrucción 3

.

.

Instrucción n

Estructuras de programación

■ Lineal o secuencial

Las instrucciones se ejecutan una a una en el orden en que están escritas

Pelar patatas

Cortarlas a cuadraditos

.....

■ Repetitiva o cíclica

Un grupo de instrucciones se ejecutará un número determinado de veces

Repetir

Dejar que cuaje

Dar vuelta

Hasta que esté a nuestro gusto

■ Alternativa

Se ejecutara un grupo u otro de instrucciones según alguna condición especificada.

Si tenemos de todo

Pelar patatas

.....

Si no

Mensaje: “Debe comprar todos los ingredientes y utensilios”

Finsi

Formas de expresar algoritmos

- **Diagramas de flujo**
- **Pseudocódigo**
- **Otras,...**

ALGORITMOS

Ej: Sumar y hallar la media de dos números y escribir en pantalla los resultados.

Pseudocódigo



.....
Inicio

leerNumeros

calcularSuma

calcularMedia

escribirResultados

Fin

.....
Inicio

//leerNumeros

Leer (numero1, numero2)

//calcularSuma

suma = numero1 + numero2

//calcularMedia

media = suma /2

//escribirResultados

Escribir ("Los resultados son:"
suma, media)

Fin

Pasos para resolución de algoritmos

1. Análisis del problema: Escritura detallada de las especificaciones
2. Diseño del programa: Diseñar los tipos (si procede) y escribir en **pseudocódigo** el algoritmo principal, dividiéndolo en módulos.
3. Escribir en pseudocódigo los módulos restantes si los hay.
4. Prueba
 - ❑ Revisión visual del pseudocódigo
 - ❑ Seguir la traza con juego de ensayo
5. Optimización del algoritmo
6. Documentación



Ver pag.22

Pasos para resolución de algoritmos

6. Documentación

- ❑ Externa
- ❑ Interna
 - **Comentarios en línea:** /*Esto es un ejemplo de comentario en línea*/
 - **Código autodocumentado:** nombres alusivos a los objetos que se usen.
 - **Impresión agradable**

Ver pag.24

ALGO- RIT- MOS: pseudo- código

PROGRAMA: Nombre_del_programa.

Comentario: *//Documentación externa*

ANÁLISIS:

//Discusión y comprensión del problema.

Propósito: *Breve descripción de lo que realiza.*

Entrada: *datos de entrada al programa.*

Salida: *información de salida del programa.*

Suposiciones: *Todas aquellas que vayan a configurar el entorno en el que se desarrolla el algoritmo.*

ENTORNO:

CONSTANTES: Declaraciones e inicializaciones. Comentario

VARIABLES: Declaraciones e inicializaciones. Comentario

OTROS OBJETOS:

Declaraciones de los módulos, si los hay u otros objetos que vayamos a usar (los veremos en capítulos posteriores).

Programa principal

Inicio

<instrucciones que forman el programa> *//Comentario: documentación interna*
<llamadas a los módulos, si los hubiera>

Fin del Programa Principal

NOMBRE_MODULO_1

Inicio

<instrucciones que forman el módulo>

Fin NOMBRE _MODULO_1

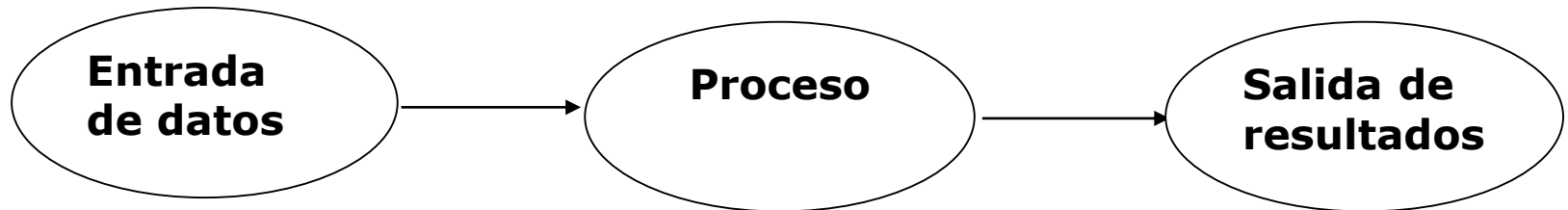
NOMBRE_MODULO_N

Inicio

<instrucciones que forman el módulo>

Fin NOMBRE _MODULO_N

Proceso general de un programa



INSTRUCCIONES DE ENTRADA

Formato:

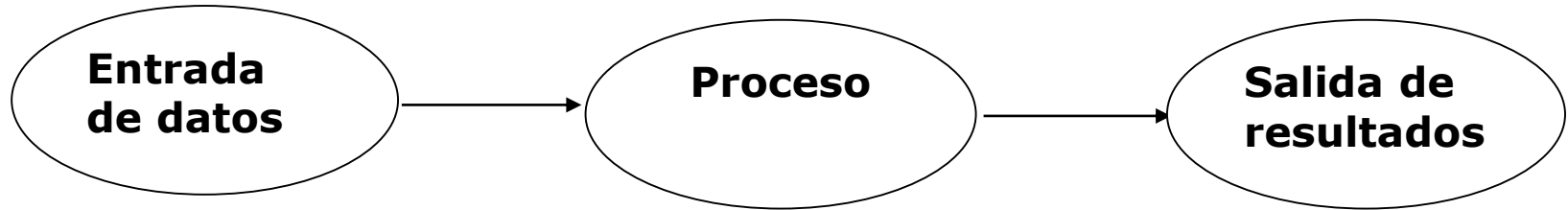
Leer (Nombres_Variables)

Ver ejemplos pag.30

- 1- Obtener dato del terminal (teclado)
 - 2- Almacenar dato en dirección referenciada por el **nombre de la variable** correspondiente
- Repetir 1 y 2 mientras queden variables en la lista



Proceso general de un programa

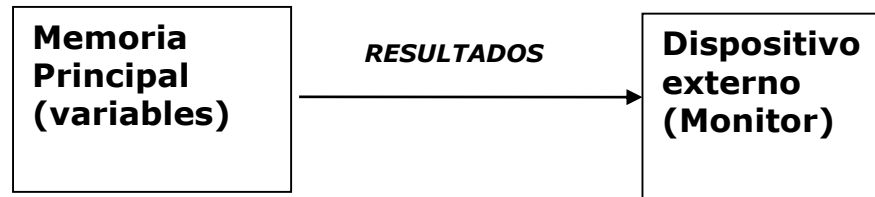


INSTRUCCIONES DE SALIDA

Formato:

Escribir (mensaje, Nombres_Variables)

- 1- Escribir **mensaje** en pantalla si lo hay
 - 2- Si hay variables
 - 3- Escribir **variable** referenciada en pantalla
- Repetir 3 mientras queden variables en la lista



“Las cosas no son difíciles de hacer, lo
que es difícil es ponerse a hacerlas.”

Constantin Brancusi

(Escultor, Rumania, 1876-1957)