PROGRAMACIÓN Y MÉTODOS NUMÉRICOS
2503506

ESTRUCTURAS DE DATOS

Andrés Agudelo
Departamento de Ingeniería Mecánica
andres.agudelos@udea.edu.co

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Contenido

1. Introducción
2. Cadenas de caracteres
   - Instrucciones
   - Operaciones
3. Arreglos
   - Arreglos unidimensionales
   - Arreglos bidimensionales
   - Arreglos multidimensionales
4. Registros
5. A continuación

1. Introducción
2. Cadenas de caracteres
   - Instrucciones
   - Operaciones
3. Arreglos
   - Arreglos unidimensionales
   - Arreglos bidimensionales
   - Arreglos multidimensionales
4. Registros
5. A continuación

Introducción

Estructuras de datos

Estructura de datos → Colección de datos que se pueden caracterizar por su organización y por las operaciones que se pueden realizar con ella.

Principales tipos de datos:

- **Simples:**
  - Entero (`int`)
  - Real (`float`)
  - Complejo (`complex`)
  - Carácter (`str`)
  - Lógico (`bool`)
- **Estructurados:**
  - Cadenas de caracteres (`str`)
  - Arreglos (array: `numpy.ndarray`)
  - Registros (`tuple`, `list`, `dict`, DataFrame)

1 Introducción

2 Cadenas de caracteres
- Instrucciones
- Operaciones

3 Arreglos
- Arreglos unidimensionales
- Arreglos bidimensionales
- Arreglos multidimensionales

4 Registros

5 A continuación

Cadenas de caracteres

Cadenas de caracteres

Una cadena de caracteres (`string`) es un conjunto de caracteres, incluido el espacio, que se almacenan en un área contigua de la memoria.

Pueden ser entradas o salidas desde/hacia un terminal.

Características:

- Longitud → Cantidad de caracteres que contiene (entero).
- Sintaxis → Comas simples ' ' (Python: “, ”).
- Instrucciones básicas → Asignación, entrada, salida.
- Operaciones → Longitud, comparación, concatenación, conversión cadena/número (+ propias de cada lenguaje).
- Codificación → Cada carácter es diferente (`m` $\neq$ `M`): Código ASCII (*American Standard Code for Information Interchange*).

Cadenas de caracteres

Valor ASCII	Carácter	Valor ASCII	Carácter	Valor ASCII	Carácter	Valor ASCII	Carácter
000	NUL	032	espacio	064	@	096	'
001	SOH	033	!	065	A	097	a
002	STX	034	"	066	B	098	b
003	ETX	035	#	067	C	099	c
004	EOT	036	$	068	D	100	d
005	ENQ	037	%	069	E	101	e
006	ACK	038	&	070	F	102	f
007	BEL	039	'	071	G	103	g
008	BS	040	(	072	H	104	h
009	HT	041	)	073	I	105	i
010	LF	042	*	074	J	106	j
011	VT	043	+	075	K	107	k
012	FF	044	,	076	L	108	l
013	CR	045	-	077	M	109	m
014	SO	046	.	078	N	110	n
015	SI	047	/	079	O	111	o
016	DLE	048	0	080	P	112	p
017	DC1	049	1	081	Q	113	q
018	DC2	050	2	082	R	114	r
019	DC3	051	3	083	S	115	s
020	DC4	052	4	084	T	116	t
021	NAK	053	5	085	U	117	u
022	SYN	054	6	086	V	118	v
023	ETB	055	7	087	W	119	w
024	CAN	056	8	088	X	120	x
025	EM	057	9	089	Y	121	y
026	SUB	058	:	090	Z	122	z
027	ESC	059	;	091	[	123	{
028	FS	060	<	092	\	124	|
029	GS	061	=	093	]	125	}
030	RS	062	>	094	↑	126	~
031	US	063	?	095	_	127	DEL

NOTA: Los 32 primeros caracteres y el último son caracteres de control; no son imprimibles.

Cadenas de caracteres

Instrucciones

- **Asignación:**
  `var string: Nombre`
  `Nombre` ← `'Luis Carlos'`
- **Lectura/escritura:**
  `leer Nombre`
  `escribir Nombre`

		L	u	i	s		C	a	r	l	o	s
Elementos	→	1	2	3	4	5	6	7	8	9	10	11
Posición	→	0	1	2	3	4	5	6	7	8	9	10

Longitud = 11

Cadenas de caracteres

Operaciones

Cálculo de la longitud:

longitud ⇒ Valor numérico

string_1 ← 'Don Quijote de la Mancha'

longitud(string_1) ⇒ 24

longitud('Don Quijote de la Mancha') ⇒ 24

L ← longitud('precios')

$3 * L - 1 \Rightarrow 20$

$4 + 5 +$ longitud('DEMO') $\Rightarrow 4 + 5 + 4 \Rightarrow 13$

Cadenas de caracteres

Operaciones

Comparación (Igualdad):

Dos cadenas de caracteres a y b, de longitudes m y n son iguales si se cumplen dos condiciones:

1. El número de caracteres de a y b son los mismos (m = n).
2. Cada carácter de a es igual a su correspondiente de b:
   Si $a = a_1, a_2, \ldots a_n$, y $b = b_1, b_2, \ldots b_n$, se debe verificar que $a_i = b_i$ para todo *i* en el rango $1 \leq i \leq n$.

Ejemplo:

'EMILIO' == 'EMILIO' ⇒ verdadero
'EMILIO' == 'Emilio' ⇒ falso
'EMILIO' == 'EMILIA' ⇒ falso
'EMILIO' == 'EMILIO ' ⇒ falso

Cadenas de caracteres

Operaciones

Concatenación:

Unión de varias cadenas de caracteres en una sola ⇒ &

'MIGUEL' & 'DE' & 'CERVANTES' ⇒ 'MIGUELDECERVANTES'

'MIGUEL ' & 'DE ' & 'CERVANTES' ⇒ 'MIGUEL DE CERVANTES'

'MIGUEL' & ' DE ' & 'CERVANTES'⇒ 'MIGUEL DE CERVANTES'

'MIGUEL' & ' ' & 'DE' & ' ' & 'CERVANTES'⇒ 'MIGUEL DE CERVANTES'

var string: Nombre, espacio, Apellido
Nombre ← 'Carlos'
espacio ← ' '
Apellido ← 'Fuentes'

Nombre & espacio & Apellido ⇒ 'Carlos Fuentes'

Cadenas de caracteres

Operaciones

Búsqueda:

Recorrido por los elementos de una cadena de caracteres ⇒ índice

El índice toma el valor de la posición de cada carácter dentro de la cadena.

var string: Nombre

Nombre ← 'Carlos'

longitud(Nombre) ⇒ 6 (índice → 0, 1, 2 ,3, 4, 5)

Nombre(2) ⇒ 'r'

Nombre(2) & Nombre(4) & Nombre(5) & Nombre(1) ⇒ 'rosa'

Cadenas de caracteres

Operaciones

Conversión cadena/número:

Convierte un tipo de datos en otro ⇒ `string_a_numero`, `numero_a_string`

Conversión bidireccional entre ambos tipos de datos.

```
var  string:  Nombre      real:  Numero
Nombre ← '286'
Numero ← 1350 + 150   ⇒   1500

Nombre + 14 ⇒ Error
string_a_numero(Nombre) + 14   ⇒   300
Numero_1 ← numero_a_string(Numero)   ⇒   '1500'
Numero_1(1) & Numero_1(0)   ⇒   '51'
```


1 Introducción

2 Cadenas de caracteres
- Instrucciones
- Operaciones

3 Arreglos
- Arreglos unidimensionales
- Arreglos bidimensionales
- Arreglos multidimensionales

4 Registros

5 A continuación

Arreglos (*arrays*)

Arreglos

Son agrupaciones finitas de datos organizados y homogéneos.

- **Finito**:
  Se debe establecer un límite para el número de elementos que contendrá.
- **Organizado**:
  Cada elemento se puede determinar según su índice o posición dentro del arreglo.
- **Homogéneo**:
  Todos los datos en un arreglo deben ser del mismo tipo (numéricos, alfanuméricos, etc.). Excepción ⇒ Registros.

Arreglos

Atributos

- **Componentes:**
  Elementos que conforman el arreglo ⇒ valores que se almacenan en cada una de las ubicaciones de éste.
- **Índice:**
  Variable, constante o expresión que debe resultar en un número natural.

  ⇒ Indica la posición correspondiente a un elemento en el arreglo.

  ⇓

  Se usa tanto para efectos de lectura/escritura, como de asignación (incluyendo el cero).

Arreglos

Atributos

Sintaxis:

`arreglo_1` ⇒ Se refiere a todos los elementos de la variable `arreglo_1`.

`arreglo_1(k)` ⇒ Se refiere al elemento ubicado en la posición `k` de la variable `arreglo_1`.

- Lectura: `x` ← `arreglo_1(k)`.
- Escritura: **`escribir`** `arreglo_1(k)`.
- Asignación: `arreglo_1(k)` ← 5.

Arreglos unidimensionales – Vectores

Arreglos unidimensionales

Se conocen comúnmente como vectores.

⇓

Colección o conjunto de valores numéricos ordenados en forma de lista.

Vector:

- Es un arreglo de una sola dimensión. ⇒ Requiere sólo un índice para el posicionamiento de sus elementos.
- Se puede presentar como una fila o una columna de datos.

Declaración:

var arreglo: `Vector_1`(1, n), `Vector_2`(m, 1)

Vectores

var arreglo: `X`(1, 8)

	X(0)	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	X(7)
Elemento ⇒	14.0	12.0	8.0	7.0	6.41	5.23	6.15	7.25
Índice ⇒	0	1	2	3	4	5	6	7

`i` ← 4

X(i+1) ⇒ Representa el elemento X(5), de valor 5.23

X(i−2) ⇒ Representa el elemento X(2), de valor 8.0

X(i+7) ⇒ Representa el elemento X(11): Error

X(i−5) ⇒ Representa el elemento X(-1): Error

Excepción: Python

Vectores

X(0)	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	X(7)
14.0	12.0	8.0	7.0	6.41	5.23	6.15	7.25

Operaciones básicas con vectores

Acción	Resultado
`escribir X(0)`	Muestra el valor de X(0) ⇒ 14.0
`leer X`	Carga el vector X con sus 8 elementos
`X(3)` ← 45	Actualiza el valor de X(3) a 45
`SUMA` ← `X(0)` + `X(2)`	Almacena el valor de `X(0)` + `X(2)` = 22 en la variable `SUMA`
`SUMA` ← `SUMA` + `X(3)`	Añade el valor de `X(3)` a la variable `SUMA` ⇒ `SUMA` = 67
`X(4)` ← `X(4)` + 3.5	Suma 3.5 al valor de `X(4)` ⇒ `X(4)` pasa a ser 9.91
`X(5)` ← `X(6)` − `X(1)`	Actualiza el valor de `X(5)` a `X(6)`−`X(1)` = −5.85

Vectores

Recorrido de vectores

Efectuar una acción general sobre todos los elementos de un vector.

$\Downarrow$

Estructuras repetitivas (**para**), cuyas variables de control (por ejemplo, `i`) se utilizan como índices del vector.

algoritmo Recorrido de un vector

```
Entradas:     arreglo:  Vector_1(1, n)     %Vector fila con n valores
Intermedias:  entero:   i                  %Índice
                        n                  %Longitud del vector
inicio
    leer Vector_1
    n ← longitud(Vector_1)
    para i = 0 hasta n − 1 haga
        Instrucciones que involucran Vector_1(i)
    fin para
fin
```


Vectores

Ejemplo 1

Dado un vector fila de 15 elementos, diseñe un algoritmo que le reste a cada elemento un valor igual al 10 % del índice correspondiente a su posición.

Análisis:
Se debe conocer el vector (`Vector_1`), y recorrerlo con una estructura de repetición (`para`). En cada iteración se procesará un elemento del vector, actualizando su valor según el valor actual del índice (`i`):

$$\texttt{Vector\_1(i)} \leftarrow \texttt{Vector\_1(i)} - 0.1 * \texttt{i}$$

Vectores

algoritmo Incremento selectivo de un vector

```
Entradas:     arreglo:  Vector_1(1, 15)     %Vector fila con 15 valores
Intermedias:  entero:   i        %Variable entera para recorrer el vector
                        n                  %Longitud o tamaño del vector
inicio
    leer Vector_1
    n ← longitud(Vector_1)
    para i = 0 hasta n − 1 haga
        Vector_1(i) ← Vector_1(i) − 0.1 * i
    fin para
    escribir Vector_1
fin
```


Vectores

Ejemplo 2

Calcular la estatura media de los alumnos de una clase, y determinar cuántos están por encima (altos) y cuántos por debajo de ésta (bajos).

Realizar la prueba de escritorio con el siguiente vector de estaturas:

$$\texttt{estaturas} = [\ 1.60 \quad 1.56 \quad 1.75 \quad 1.58 \quad 1.52 \quad 1.82 \quad 1.67 \quad 1.74 \quad 1.69 \quad 1.71\ ]$$

Análisis

Se deberá recorrer el vector `estaturas` usando un contador (`i`) para acumular éstas en una variable (`suma`), la cual luego permitirá calcular la media (`media`).

Después se deberá recorrer de nuevo el vector `estaturas` para clasificar a los alumnos según su estatura (`altos` y `bajos`).

Vectores

Ejemplo 2 – Análisis

- **Entradas:**
  arreglo: `estaturas(1, n)` → Vector de estaturas
- **Salidas:**
  real: `media` → Estatura media de los alumnos
  entero: `altos` → Número de estudiantes con estatura sobre la media
  `bajos` → Número de estudiantes con estatura bajo la media
- **Intermedias:**
  entero: `n` → Número total de alumnos
  `i` → Contador de alumnos
  real: `suma` → Acumulador de estaturas

⇒ **Se requieren dos estructuras de repetición.**

Vectores

algoritmo Clasificación de estaturas

```
var      real:   n, i, suma, media, altos, bajos      arreglo:   estaturas(1, n)
1  inicio
2      suma ← 0;      bajos ← 0;      altos ← 0
3      leer estaturas
4      n ← longitud(estaturas)
5      para i = 0 hasta n − 1 haga
6          suma ← suma + estaturas(i)
7      fin para
8      media ← suma / n
9      para i = 0 hasta n − 1 haga
10         si estaturas(i) < media entonces
11             bajos ← bajos + 1
12         sino
13             si estaturas(i) > media entonces
14                 altos ← altos + 1
15             fin si
16         fin si
17     fin para
18     escribir 'Estatura media: ', media, ' [m]'
19     escribir 'No. de estudiantes con estatura por encima de la media: ', altos
20     escribir 'No. de estudiantes con estatura por debajo de la media: ', bajos
21 fin
```


Vectores

`estaturas` = [1.60 1.56 1.75 1.58 1.52 1.82 1.67 1.74 1.69 1.71]

n	i	estaturas(i)	suma	media	bajos	altos	Pantalla
			0		0	0	
10			0		0	0	
10	0	1.6	1.6		0	0	
10	1	1.56	3.16		0	0	
10	2	1.75	4.91		0	0	
10	3	1.58	6.49		0	0	
10	4	1.52	8.01		0	0	
10	5	1.82	9.83		0	0	
10	6	1.67	11.5		0	0	
10	7	1.74	13.24		0	0	
10	8	1.69	14.93		0	0	
10	9	1.71	16.64		0	0	
10	10	1.71	16.64	1.664	0	0	
10	0	1.6	16.64	1.664	1	0	
10	1	1.56	16.64	1.664	2	0	
10	2	1.75	16.64	1.664	2	1	
10	3	1.58	16.64	1.664	3	1	
10	4	1.52	16.64	1.664	4	1	
10	5	1.82	16.64	1.664	4	2	
10	6	1.67	16.64	1.664	4	3	
10	7	1.74	16.64	1.664	4	4	
10	8	1.69	16.64	1.664	4	5	
10	9	1.71	16.64	1.664	4	6	
10	10	1.71	16.64	1.664	4	6	'Estatura media: 1.664 [m]'
10	10	1.71	16.64	1.664	4	6	'No. de estudiantes con estatura por encima de la media: 6'
10	10	1.71	16.64	1.664	4	6	'No. de estudiantes con estatura por debajo de la media: 4'

Arreglos bidimensionales – Matrices

Arreglos bidimensionales

Se conocen comúnmente como matrices (tablas).

- Son un arreglo de vectores ⇒ Dos dimensiones (`m,n`)
  `m` → Número de filas
  `n` → Número de columnas
  $\texttt{m} > 1$ y $\texttt{n} > 1$
- Se requieren dos índices para el posicionamiento de los elementos en el arreglo: (i, j).
  i → Especifica la fila $(0 \leq i \leq \texttt{m} - 1)$
  j → Especifica la columna $(0 \leq j \leq \texttt{n} - 1)$
- Se presentan como una tabla de datos.

Declaración:

var `arreglo: Matriz_1(m, n)`

Matrices

$$\texttt{Matriz\_1} = \begin{array}{|c|c|c|}\hline 23 & 78 & 100 \\ \hline 67 & 99 & 45 \\ \hline 1 & 74 & 325 \\ \hline 22 & 10 & 20 \\ \hline \end{array} \qquad \texttt{m} = 4 \qquad \texttt{n} = 3$$

`Matriz_1(1,2)` ⇒ 45; `Matriz_1(3,0)` ⇒ 22

`i` ← 1; `j` ← 0
`Matriz_1(i−1,j+2)` ⇒ `Matriz_1(0,2)` ⇒ 100
`Matriz_1(i+3,j)` ⇒ `Matriz_1(4,0)` ⇒ Error
`Matriz_1(i,j−3)` ⇒ `Matriz_1(1,−3)` ⇒ Error

tamaño(`Matriz_1`) ⇒ [4, 3] → Vector de dos elementos
`tam_M` ← **tamaño**(`Matriz_1`): `tam_M(0)` ⇒ 4
tamaño(`Matriz_1`)(1) ⇒ 3

Matrices

Recorrido de matrices

Consiste en pasar por todos los elementos de un arreglo bidimensional. Se requieren dos estructuras repetitivas anidadas, cuyas variables de control (por ejemplo, `i, j`) se utilizan como índices de la matriz ⇒ Filas y columnas.

algoritmo Lectura de los elementos de una matriz

```
Entradas:    arreglo: Matriz_1(m, n)          %Matriz de dimensión (m × n)
Intermedias: entero:  i, j                    %Variables enteras para recorrer la matriz
1 inicio
2     leer m, n
3     para i = 0 hasta m − 1 haga
4         para j = 0 hasta n − 1 haga
5             leer Matriz_1(i, j)
6         fin para
7     fin para
8 fin
```

¿Lectura por columnas o por filas?

Matrices

Ejemplo 3

Dada una matriz `M`, de dimensión `(m,n)`, desarrollar un algoritmo que identifique la cantidad de valores positivos y negativos en ésta, y además calcule la suma de todos los valores positivos, y la de todos los valores negativos.

Análisis:
Se debe conocer la matriz, y de ahí saber cuál es su tamaño o dimensión. Luego se debe recorrer ésta elemento por elemento (primero en una dimensión, y luego en la otra), evaluando cada elemento, acumulando las respectivas sumas, y contando los valores positivos y negativos.

Matrices

Ejemplo 3 – Análisis

- **Entradas:**
  arreglo: `M` ⇒ Matriz
- **Salidas:**
  real: `suma_pos` ⇒ Suma de los números positivos en la matriz
  `suma_neg` ⇒ Suma de los números negativos en la matriz
  entero: `num_pos` ⇒ Cantidad de números positivos en la matriz
  `num_neg` ⇒ Cantidad de números negativos en la matriz
- **Intermedias:**
  entero: `m, n` ⇒ Número de filas y de columnas de la matriz `M`
  `i, j` ⇒ Índices para recorrer la matriz

Matrices

algoritmo Recorrido de una matriz

```
var     arreglo:    M(m,n)
        entero:     m, n, i, j, num_pos, num_neg
        real:       suma_pos, suma_neg
inicio
    leer M
    m ← tamaño(M)(0);    n ← tamaño(M)(1)
    num_pos ← 0;    num_neg ← 0;    suma_pos ← 0;    suma_neg ← 0
    para i = 0 hasta m − 1 haga
        para j = 0 hasta n − 1 haga
            si M(i,j) > 0 entonces
                num_pos ← num_pos + 1
                suma_pos ← suma_pos + M(i,j)
            sino
                si M(i,j) < 0 entonces
                    num_neg ← num_neg + 1
                    suma_neg ← suma_neg + M(i,j)
                fin si
            fin si
        fin para
    fin para
    escribir 'La suma de los ', num_pos, ' números positivos es: ', suma_pos
    escribir 'La suma de los ', num_neg, ' números negativos es: ', suma_neg
fin
```


Matrices

Ejemplo 3

Realice la prueba de escritorio con la siguiente matriz:

$$\mathtt{M} = \begin{bmatrix} 3 & -2 & 8 & -9 \\ 5 & 14 & -0.5 & 1 \\ -4 & 7 & 22.3 & 6.7 \end{bmatrix}$$

$\mathtt{m} = 3 \qquad \mathtt{n} = 4$

Matrices

m	n	i	j	M(i,j)	num_pos	suma_pos	num_neg	suma_neg	Pantalla
3	4								
3	4				0	0	0	0	
3	4	0	0	3	1	3	0	0	
3	4	0	1	-2	1	3	1	-2	
3	4	0	2	8	2	11	1	-2	
3	4	0	3	-9	2	11	2	-11	
3	4	0	4	-9	2	11	2	-11	
3	4	1	0	5	3	16	2	-11	
3	4	1	1	14	4	30	2	-11	
3	4	1	2	-0.5	4	30	3	-11.5	
3	4	1	3	1	5	31	3	-11.5	
3	4	1	4	1	5	31	3	-11.5	
3	4	2	0	-4	5	31	4	-15.5	
3	4	2	1	7	6	38	4	-15.5	
3	4	2	2	22.3	7	60.3	4	-15.5	
3	4	2	3	6.7	8	67	4	-15.5	
3	4	2	4	6.7	8	67	4	-15.5	
3	4	3	4	6.7	8	67	4	-15.5	
3	4	3	4	6.7	8	67	4	-15.5	La suma de los 8 números positivos es 67
3	4	3	4	6.7	8	67	4	-15.5	La suma de los 4 números negativos es -15.5

Arreglos multidimensionales

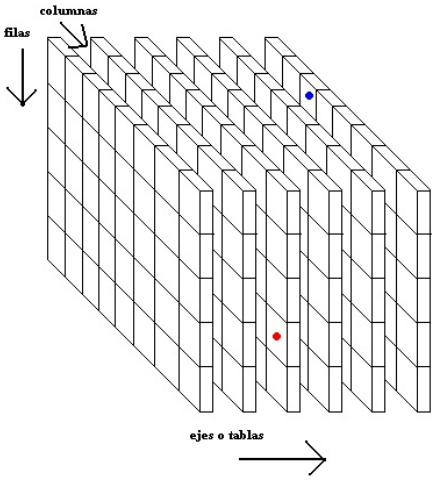
Arreglos multidimensionales

Son arreglos con más de dos dimensiones.

- Los más comunes son aquellos de tres dimensiones:
  Son un arreglo de matrices (hipermatrices) → Tres dimensiones (m,n,p)
  m → Número de filas
  n → Número de columnas
  p → Número de matrices
  $\mathtt{m} > 1 \qquad$ y $\qquad \mathtt{n} > 1 \qquad$ y $\qquad \mathtt{p} > 1$
- Requieren tres índices para el posicionamiento de los elementos en el arreglo (i,j,k).
  i → Especifica la fila $(0 \leq \mathtt{i} \leq \mathtt{m} - 1)$
  j → Especifica la columna $(0 \leq \mathtt{j} \leq \mathtt{n} - 1)$
  k → Especifica la matriz $(0 \leq \mathtt{k} \leq \mathtt{p} - 1)$

Declaración: **var** arreglo: Arreglo_1(m, n, p)

Arreglos multidimensionales



Arreglos multidimensionales

- Obedecen las mismas reglas de asignación y de operaciones que los vectores y matrices.
- Por tener tres índices, requieren de tres estructuras de repetición para su recorrido.

Arreglos multidimensionales

Ejemplo 4

Una aerolínea gestiona su sistema de reservas mediante un arreglo de tres dimensiones, en el cual las filas corresponden a la fila dentro del avión, las columnas al número del asiento, y cada matriz corresponde a un número de vuelo.

Cada elemento del arreglo tiene un valor de 1 (asiento disponible) ó 0 (asiento ocupado).

El usuario selecciona vuelo, fila y número de asiento a partir de un menú de opciones preestablecido, de modo que no hay riesgo de que se equivoque al realizar la selección.

Diseñe un algoritmo que informe a los clientes sobre la disponibilidad de un asiento solicitado, y actualice la base de datos según la selección de asiento .

Arreglos multidimensionales

Ejemplo 4 – Análisis

Se requiere conocer la información de disponibilidad de asientos (arreglo), así como el número de vuelo que elije el cliente, y el número de la fila y del asiento que se desea seleccionar.

- **Entradas:**
  - arreglo: `Reservas` ⇒ Arreglo de 3 dimensiones
  - entero: `num_vuelo` ⇒ Número de vuelo solicitado
  - `num_fila` ⇒ Número de fila solicitada
  - `num_asiento` ⇒ Número de asiento solicitado
- **Intermedias:**
  - entero: `m, n, p` ⇒ Dimensiones del arreglo
  - `i, j, k` ⇒ Índices para recorrer el arreglo
- **Salidas:**
  Mensaje que informa sobre la disponibilidad del asiento seleccionado.

Arreglos multidimensionales

algoritmo Recorrido de un arreglo tridimensional

```
   var  arreglo:  Reservas(m, n, p)      entero:  m, n, p, i, j, k, num_vuelo, num_fila, num_asiento
1  inicio
2      leer Reservas, num_vuelo, num_fila, num_asiento
3      m ← tamaño(Reservas)(0);   n ←tamaño(Reservas)(1);   p ← tamaño(Reservas)(2)
4      para k = 0 hasta p − 1 haga
5          si (k == num_vuelo) entonces
6              para i = 0 hasta m − 1 haga
7                  si (i == num_fila) entonces
8                      para j = 0 hasta n − 1 haga
9                          si (j == num_asiento) entonces
10                             si Reservas(i,j,k) == 1 entonces
11                                 escribir 'El asiento seleccionado está disponible'
12                                 Reservas(i,j,k) ← 0      %Cambio de estado de reserva
13                             sino
14                                 escribir 'El asiento seleccionado está ocupado'
15                             fin si
16                         fin si
17                     fin para
18                 fin si
19             fin para
20         fin si
21     fin para
22 fin
```

1 Introducción

2 Cadenas de caracteres
- Instrucciones
- Operaciones

3 Arreglos
- Arreglos unidimensionales
- Arreglos bidimensionales
- Arreglos multidimensionales

4 Registros

5 A continuación

Registros

Registros

Son una estructura que almacena diferentes tipos de datos en una misma variable.

- Es una estructura heterogénea.
- Puede almacenar datos numéricos y alfanuméricos al mismo tiempo.
- Puede ser un arreglo multidimensional de estructuras homogéneas → Cada elemento es, o un arreglo, o una cadena de caracteres.

```
registro Empleado
  string:  nombre
  entero:  idNumero
  real:    salario
  fecha:   fecha_nacimiento
  entero:  antiguedad
fin registro
```

```
Empleado(2)
'Pedro'
003
5.000.000
19/04/1975
16
```

Python → Diccionarios, listas, tuplas, DataFrames

Registros

Ejemplo 5

Una empresa organiza el registro de nómina para todos sus empleados usando el código de cada uno, su nombre, y el salario correspondiente.

Almacenar cada uno de estos tipos de datos en un arreglo independiente es poco eficiente. Por lo tanto, se puede recurrir a crear un registro que contenga toda la información necesaria para gestionar la nómina.

Número de empleado	Nombre del empleado	Salario
97005	Mackoy, José Luis	1.500
95758	Mortimer, Juan	1.768
87124	Rodríguez, Manuel	2.456
67005	Carrigan, Luis José	3.125
20001	Mackena, Luis Miguel	2.156
20020	García de la Cruz, Heraclio	1.990
99002	Mackoy, María Victoria	2.450
20012	González, Yiceth	4.780
21001	Verástegui, Rina	3.590
97005	Collado, Concha	3.574

A continuación

Próxima clase

Estructuras de datos en Python.