

Capítulo 1

Estructuras Fundamentales de Datos



Introducción

- Con el propósito de que la computadora procese la información esta debe ser almacenada en la memoria. De acuerdo con la forma en que los datos se organizan se clasifican en:
 - *Tipos de datos simples (TDS)
 - *Tipos de datos estructurados (TDE)



TDS

□ EJEMPLO

□ Universidad con 50 calificaciones

¿Cuántos alumnos tienen calif > PROM?

CONT Entero

PROM, AC y Ci Reales

Leer C1 .. C50

hace AC \leftarrow C1+.. C50

PROM \leftarrow AC/50

CONT \leftarrow 0

Si C1 > PROM entonces CONT \leftarrow CONT +1

fin 3

Si C2 > PROM entonces CONT \leftarrow CONT +1

fin 4

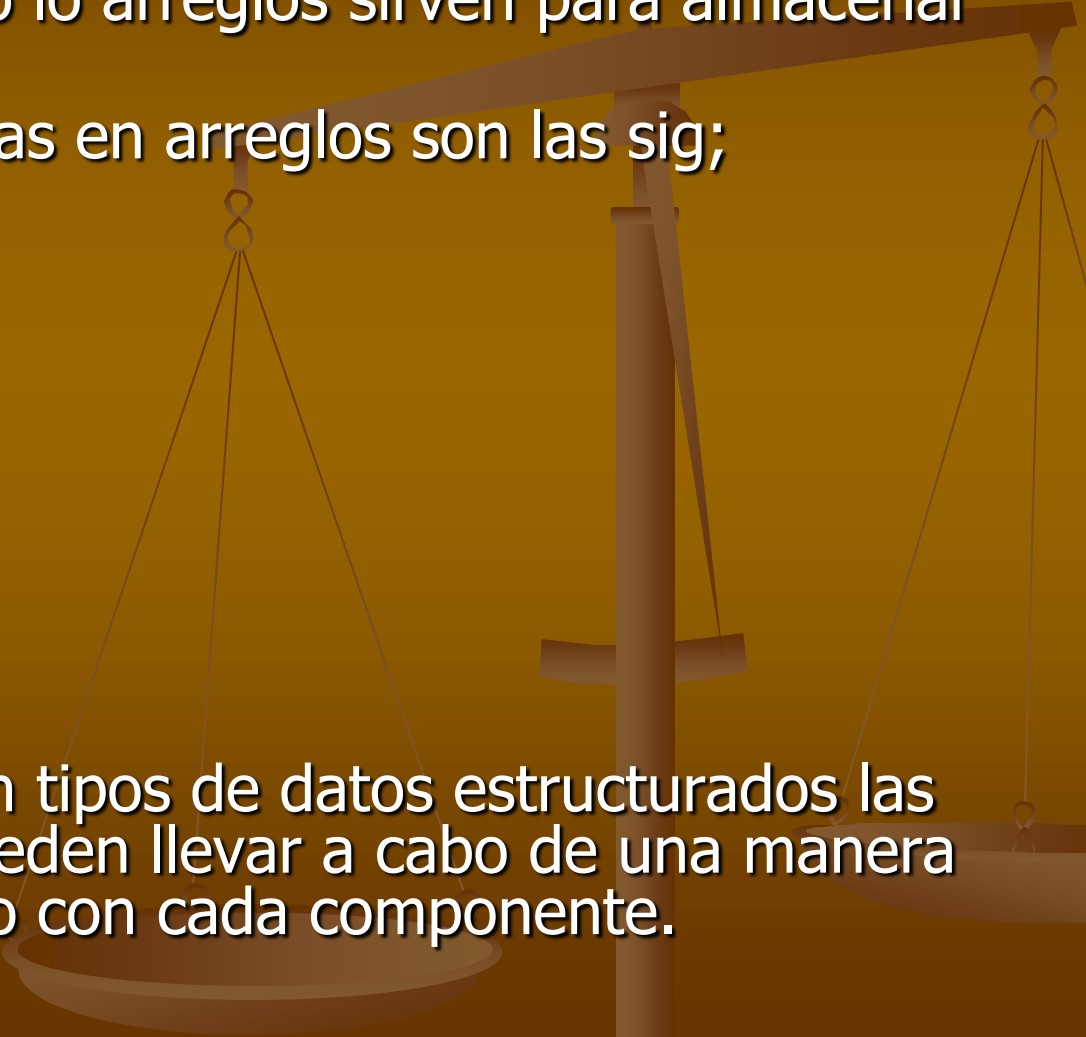
.....

100. Si C100 > PROM entonces CONT \leftarrow CONT +1

101 fin 100

102 Escribir CONT

Operaciones con arreglos unidimensionales

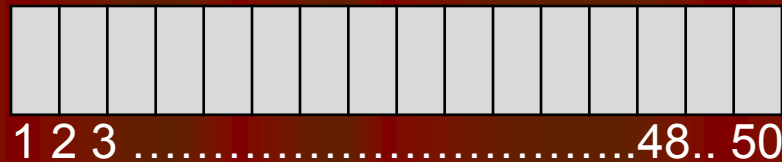
- Como ya se menciona los arreglos sirven para almacenar datos.
 - Las operaciones válidas en arreglos son las siguientes;
 - Lectura/Escritura
 - Asignación
 - Actualización:
 - Inserción
 - Eliminación
 - Modificación
 - Ordenación
 - Búsqueda
 - Como los arreglos son tipos de datos estructurados las operaciones no se pueden llevar a cabo de una manera global sino trabajando con cada componente.
- 

Lectura

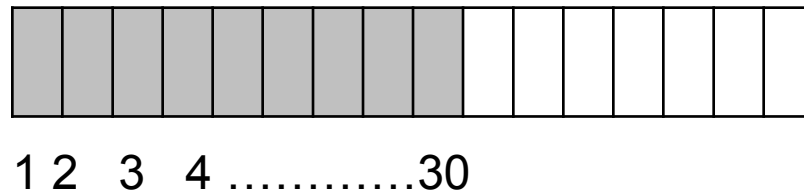
- El proceso de lectura de un arreglo consiste en leer y asignar un valor a cada uno de los componentes. Supóngase que se desea leer todos los elementos del arreglo unidimensional V en forma consecutiva.

Lectura

- Repetir con I del 1 hasta 50
- Leer $V[I]$
- Para $I=1$ se lee $V[1]$
-
- Para $I=50$ se lee $V[50]$
- Al finalizar el ciclo de lectura se tendrá asignado un valor a cada uno de los componentes del arreglo unidimensional V.



-
- Puede suceder que no se necesite leer todos los componentes
 - Repetir con I del 1 hasta 30
 - Leer $V[I]$



Escritura

- Similar a lectura
- Repetir con I desde 1 hasta N
- Escribir $V[I]$
- Para $I=1$ se escribe $V[1]$
-
- Para $I=N$ se escribe $V[N]$

Asignación

- En general no es posible asignar directamente un valor a todo el arreglo, sino que se debe asignar el valor deseado a c/componente.
- Ejemplos
- $\text{CICLO}[\text{ene}] \leftarrow 123.25$
- $\text{CICLO}[\text{mar}] \leftarrow \text{CICLO}[\text{ene}]/2$
- Se asigna 0 a todas las casillas.
- Repetir con mes desde ene hasta dic
- Hacer $\text{CICLO}[\text{mes}] \leftarrow 0$



Cont. Asignación

- Algunos lenguajes de programación es posible asignar una variable tipo arreglo a otra del mismo tipo (arreglo)

- $V1 \leftarrow V$

Es equivalente a:

- Repetir con I desde 1 hasta 50
- $V1[I] \leftarrow V[I]$

Actualización

- La actualización es una operación que se realiza frecuentemente en los arreglos. La cantidad de actualizaciones es directamente proporcional al problema que se intenta resolver. A diferencia de las otras operaciones estudiadas, la actualización lleva implícita otras operaciones como inserción y eliminación.

Cont. Actualización

- Con el propósito de realizar una actualización de manera eficiente, es importante conocer si el arreglo está o no ordenado, ie si sus componentes respectan algún orden (ascendiente o decreciente). Cabe destacar que las operaciones de inserción, eliminación y modificación serán tratadas de forma separada para arreglos ordenados y desordenados.
- Finalmente, es importante señalar que la operación de búsqueda se utiliza como auxiliar en las operaciones de inserción, eliminación y modificación. Por esta razón se presenta la búsqueda secuencial en a. desordenados. (Tema de búsquedas)

Algoritmo 1.3 Busca secuencial_desordenado

- ◆ V es un a. desordenado de 100 elementos, N es el número actual de elementos y X el valor a buscar.
- ◆ 1. Hacer $I \leftarrow 1$
- ◆ 2. Mientras $(I \leq N)$ y $(X \neq V[I])$ Repetir
 - ◆ Hacer $I \leftarrow I + 1$
- ◆ 3. Fin del ciclo paso 2
- ◆ 4. Si $I > N$ (No se encontró el valor buscado)
 - ◆ Entonces
 - ◆ Escribir "El valor X no esta en el arreglo"
 - ◆ Si no Escribir "El valor X esta en la posición I"
- ◆ 5. (fin del condicional del paso 4)
- ◆ El método es sencillo aunque no muy eficiente. Consiste en recorrer el arreglo hasta que con éxito o fracaso se encuentre el valor buscado.

Arreglos desordenados

- Considere un arreglo unidimensional de 100 elementos con N componentes asignados.
- a.1) **Inserción:** Para insertar un elemento Y en V desordenado, se debe verificar que exista espacio. Se asigna posición $N+1$.

Algoritmo 1.4 Inserta_desordenado

- V arreglo de máximo 100 elementos. No es el número actual de elementos. Y el valor a insertar.
- Si $N < 100$
- Entonces
- Hacer $N \leftarrow N+1$ y $V[N] \leftarrow Y$
- Si no
- Escribir “El valor de Y no se puede insertar. No hay espacio disponible”
- {fin del paso 1}

Eliminación

- ¿Qué se debe verificar?
Que el elemento a borrar existe.

Después?

Se recorrerán los elementos 1 casilla a la izq. Y N disminuirá en 1.

Algoritmo de Eliminación.

- Se incluirá el algoritmo 1.3 de búsqueda
- V es un a. desordenado de 100 elementos, N es el número actual de elementos y X el valor a eliminar.
- {I y K son variables de tipo entero}
- 1. Hacer $I \leftarrow 1$
- 2. Mientras $(I \leq N)$ y $(X \neq V[I])$ Repetir
 - Hacer $I \leftarrow I+1$
- 3. Fin del ciclo paso 2
- 4. Si $I > N$ (No se encontró el valor buscado)
 - Entonces
 - Escribir "El valor X no esta en el arreglo"
 - Si no Escribir "El valor X esta en la posición I"
 - 1.1 Repetir con K desde I hasta $(N-1)$
 - Hacer $V[k] \leftarrow V[k+1]$
 - 1.2 {fin del ciclo del paso 4.1}
 - Hacer $N \leftarrow N-1$
- 5. (fin del condicional del paso 4)

Modificación:

¿Qué hay que verificar?

- Algoritmo 1.6 Modifica_desordenado
- El algoritmo modifica un arreglo V de max. 100 elementos. N, X el elemento de modificar por Y. Y, I enteros.
- 1. Hacer $I \leftarrow 1$
- 2. Mientras $(I \leq N)$ y $(X \neq V[I])$ Repetir
 - Hacer $I \leftarrow I+1$
- 3. Fin del ciclo paso 2
- 4. Si $I > N$ (No se encontró el valor buscado)
 - Entonces
 - Escribir “El valor X no esta en el arreglo”
 - Si no
 - Hacer $V[I] \leftarrow Y$
- 5. {fin del condicional paso 4}

Clase 14 de agosto

- Revisión de datos abstractos
- Operaciones de arreglos ordenados
- Comienzo de arreglos bidimensionales
- Programas

Arreglos Ordenados

- Considere el arreglo unidimensional ordenado V de 100 elementos. Los primeros N componentes del mismo tienen asignado un valor. En este caso se trabajará con un arreglo ordenado de manera creciente i.e.;
- $V[1] \leq V[2] \leq V[3] \leq V[4] \leq \dots \leq V[N]$
- Cuando se trabaja con arreglos ordenados se debe evitar alterar el orden al insertar o modificar.

Inserción:

Al insertar un elemento X en un arreglo unidimensional ordenado:

- Verificar que exista espacio
- Encontrar la posición que ocupará el nuevo elemento
- Cuando se detecte la posición recorrer todos los elementos
- Se asignará el valor de X a la posición encontrada
- Cuando el valor a insertar es mayor que el último elemento del arreglo, no habrá desplazamiento.
- Generalmente, se verifica si el elemento a insertar no existe en el arreglo. De lo contrario no se lleva a cabo la inserción ya que no interesa tener información repetida.
- Antes de presentar el algoritmo de inserción en arreglos ordenados, veremos la función de búsqueda auxiliar, para arreglos ordenados, que se utilizará en el proceso de inserción y eliminación.

Algoritmo 1.7

Busca_secuencial_ordenado

- {El algoritmo busca un elemento X en un arreglo unidimensional V de N elementos que se encuentra ordenado crecientemente. POS indica la posición de X en V (positivo) o la posición que estaría X (negativo)}
- Hacer $I \leftarrow 1$
- Mientras $(I \leq N)$ y $(V[I] < X)$ Repetir
- Hacer $I \leftarrow I + 1$
- {fin del ciclo del paso 2}
- Si $(I > N)$ o $(V[I] > X)$
- Entonces
- Hacer $POS \leftarrow -I$
- Si no
- Hacer $POS \leftarrow I$
- {fin del condicional del paso 4}

Inserción en un arreglo unidimensional ordenado

- Algoritmo 1.8
- Inserta _ordenado(V, N, Y)
- {Este algoritmo inserta un elemento Y en un arreglo unidimensional Ordenado de forma creciente. La capacidad máxima del arreglo es de 100 elementos. N indica el número actual de elementos en V}
- Si (N<100)
 - Entonces
 - Llamar al algoritmo Busca_secuencial_ordenado con V, N, Y y POS
 - 1.1 Si POS>0 {El elemento fue encontrado en el arreglo}
 - Entonces
 - Escribir “El elemento ya existe”
 - Si no
 - Hacer $N \leftarrow N+1$ y $POS \leftarrow POS*(-1)$
 - 1,1,1 Repetir con I desde N hasta POS + 1
 - Hacer $V[I] \leftarrow V[I-1]$
 - 1.1.2 \leftarrow {fin del ciclo del paso 1.1.1}
 - Hacer $V[POS] \leftarrow Y$
 - 1.2 {fin del condicional del pasos 1.1}
 - Si no
 - Escribir “No hay espacio en el arreglo”
 - 2 {Fin del condicional del paso 1}

b.2) Eliminación:

- Para eliminar un elemento X de un arreglo unidimensional ordenado V se debe;
- buscar la posición al elemento a eliminar. Si el resultado de la función es un valor positivo, significa que el elemento se encuentra en V y por lo tanto se puede eliminar. De lo contrario no se realiza ninguna operación.

Algoritmo 1.9 Elimina_ordenado

- {El alg. Elimina un elemento X de un arreglo unidimensional V de N elementos que se encuentra ordenado de forma creciente}

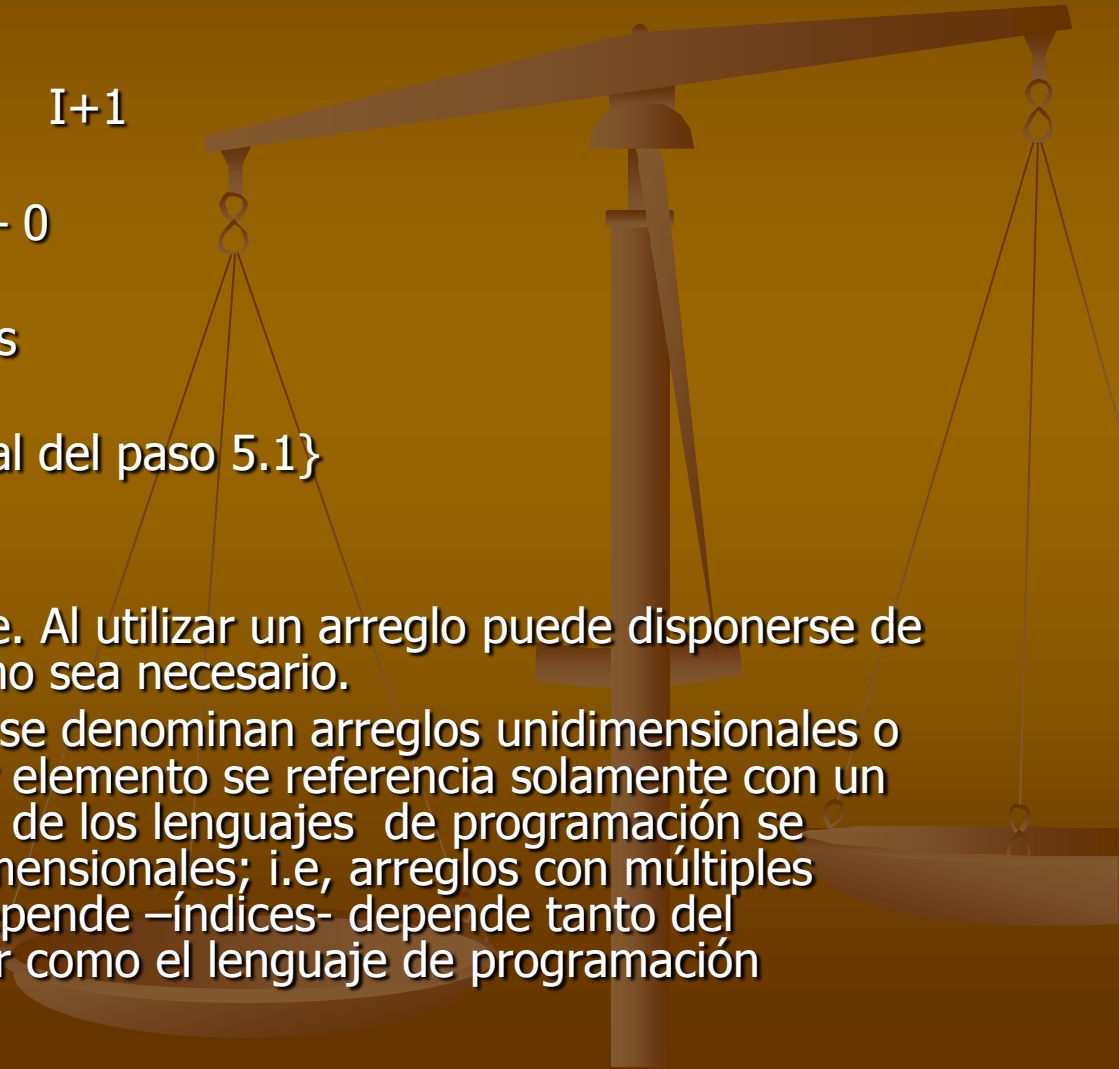
- {POS e I de tipo entero}
- Si $N > 0$
- Entonces
- Llamar el algoritmo Busca_secuencial_ordenado con V, N, X y POS
 - Si $(POS < 0)$ {no se puede eliminar}
- Entonces
- Escribir “no existe elemento”
- Si no
- Hacer $N \leftarrow N - 1$
 - Repetir con I desde POS hasta N
- Hacer $V[I] \leftarrow V[I + 1]$
- 1.1.2 {fin del ciclo del paso 1.1.1}
- 1.2 {fin del condicional del paso 1.1}
- Si no
- Escribir “El arreglo está vacío”
- {fin del condicional del paso 1}

b.3) Modificación:

- Consiste en reemplazar un componente del arreglo con otro valor.
- 1. Se busca el elemento en el arreglo
- Si se encuentra verificar que al modificar no se altere el orden
- Si se altera entonces es necesario eliminar el elemento a modificar y luego insertar el nuevo elemento en la posición correspondiente.
- Tarea

Algoritmo 1.10 Con_arreglos

- {resuelve el problema del ejemplo 1, CAL es un arreglo de 50 elementos de tipo real}
- {AC, I y CONT son de tipo entero y PROM es de tipo real}
- hacer AC ← 0
- Repetir con I desde 1 hasta 50
- Leer CAL[I]
- Hacer AC ← AC + CAL[I] e I ← I + 1
- {fin del ciclo del paso 2}
- hacer PROM ← AC/50 y CONT ← 0
- Repetir con I desde 1 hasta 50
- 5.1 Si (CAL[I] > PROM) entonces
- Hacer CONT ← CONT + 1
- 5.2 {fin del condicional del paso 5.1}
- 6. {fin del ciclo del paso 5}
- 7. Escribir CONT
- Está es la solución más eficiente. Al utilizar un arreglo puede disponerse de la información tantas veces como sea necesario.
- Los arreglos vistos hasta ahora se denominan arreglos unidimensionales o lineales, debido a que cualquier elemento se referencia solamente con un índice. Sin embargo, la mayoría de los lenguajes de programación se pueden definir arreglos multidimensionales; i.e, arreglos con múltiples índices. El # de dimensiones depende –índices- depende tanto del problema que se quiera resolver como el lenguaje de programación utilizado.



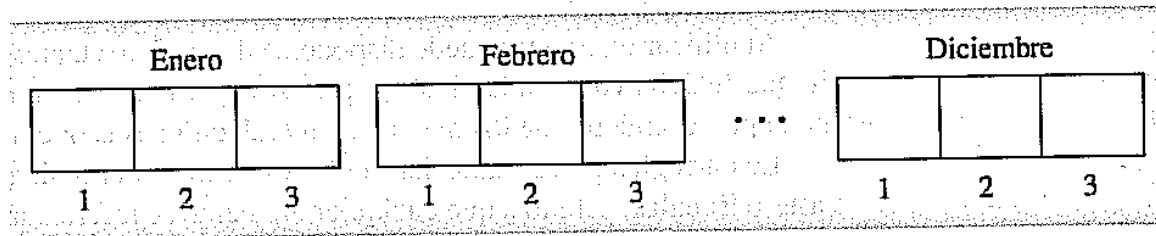
1.3 ARREGLOS BIDIMENSIONALES

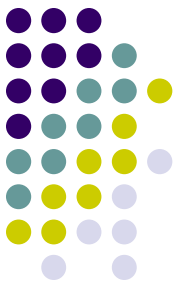
- La tabla 1.1 contiene los costos de producción de cada departamento de una fábrica, correspondiente a los 12 meses del año anterior.
 - La tabla se interpreta de la manera siguiente;
-
- Dado un mes, se conocen los costos de producción de c/u de los departamentos y dado un departamento, se conocen los costos de producción mensuales.
 - Si se quiere almacenar esta información utilizando arreglos unidimensionales, se tienen dos alternativas.

Meses/Deptos.	Dulces	Conservas	Bebidas
Enero	100	300	120
Febrero	400	200	200
Marzo	350	250	210
Abril	280	300	200
Mayo	300	320	300
Junio	250	300	350
Julio	200	280	300
Agosto	180	300	400
Septiembre	500	400	450
Octubre	350	420	220
Noviembre	400	450	360
Diciembre	600	550	531

PRIMERA OPCIÓN

1. Definir 12 arreglos de tres elementos c/u. En este caso, cada arreglo almacenará la información relativa a un mes.



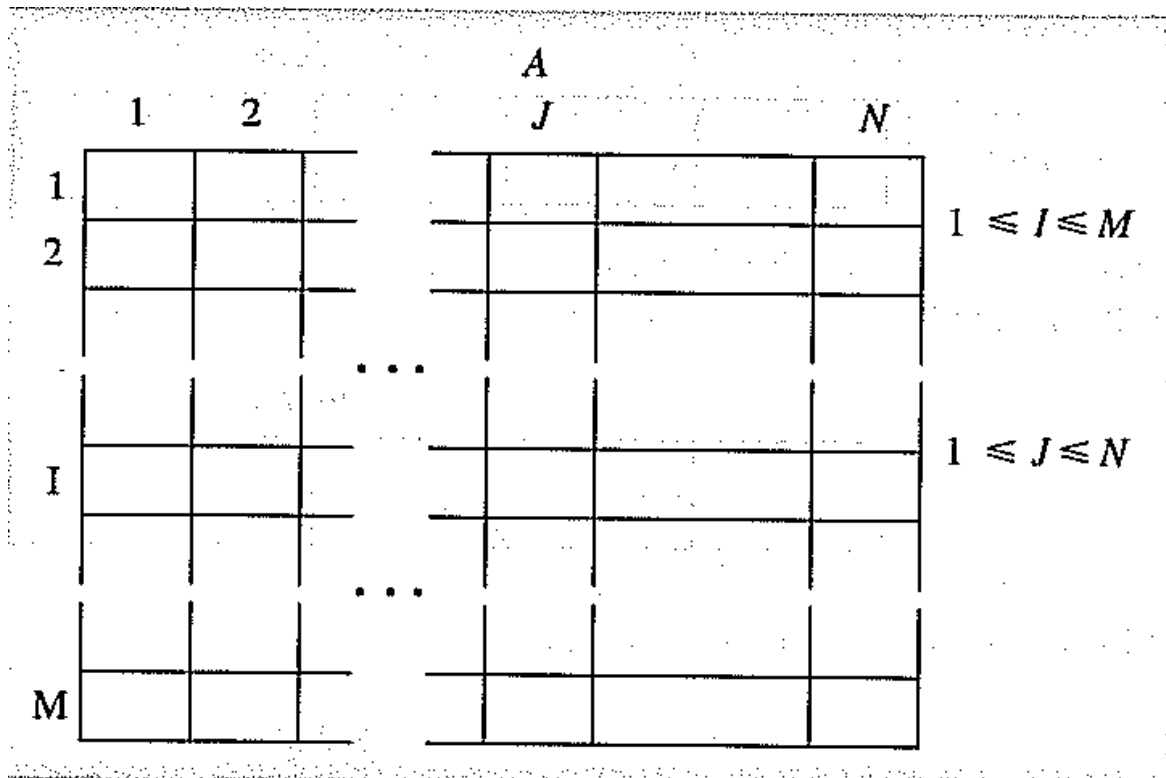


- Sin embargo, no resulta muy práctico adoptar alguna de estas dos alternativas. Se necesita una estructura que permita los datos considerando los meses –renglones y los departamentos –columnas de la tabla. Esta se denomina arreglo bidimensional.
- Un arreglo bidimensional es una colección homogénea, finita y ordenada de datos, en la que se hace referencia a cada componente del arreglo por medio de dos índices. El primero es renglón y el segundo la columnas.



Ejemplo fig 1.14

- El arreglo $A(M \times N)$ tiene M renglones y N columnas. Un elemento $A[I,J]$ se localiza en el renglón I y la columna J . Internamente en memoria se reservan $M \times N$ posiciones consecutivas.



1.3.1 Declaración de arreglos bidimensionales



- Id_arreglo =
- ARREGLO[lim_Inf_r ...lim_sup_r,
lim_Inf_c...lim_sup_c]
- NTC
- $(\text{lim_sup_r} - \text{lim_Inf_r} + 1) * (\text{lim_Inf_c} + \text{lim_Inf_c} - 1)$
- Ejemplos

1.6 Registros

- De acuerdo con lo visto en clase los arreglos son ED muy útiles para almacenar una colección de datos, todos del mismo tipo. Sin embargo en la práctica se necesitan estructuras que permitan almacenar datos de distintos tipos que sean manipulados como un único dato.
- Ejemplo de Datos de Alumno visto en clase
- Nombre, Dirección , # cuenta
- No se pueden almacenar en un arreglo. La estructura que puede guardar esta información en de manera efectiva se conoce como **registro o estructura**.
- Un registro se define como una colección heterogénea de elementos. También representa un tipo de dato estructurado, en el que cada uno de sus componentes se denomina **campo**. Los campos de un registro pueden ser todos de diferentes tipos de datos. Por lo tanto podrían ser arreglos o registros. Cada campo se identifica con un nombre único, el identificador de campo. Otra diferencia importante con los arreglos es que no hay que establecer un orden entre campos.

1.6.1 Declaración de registros

- No se seguirá la sintaxis de ningún lenguaje de programación.
- `Id_registro = REGISTRO`
- `Id_campo1 = tipo1`
- `Id_campo2 = tipo2`
- `.`
- `.`
- `Id_campon = tipon`
- {fin de la declaración del registro 1}
- Donde: `id_registro` es el nombre del dato tipo registro
- `Id_campoi` es el nombre del campo `i`
- `Id_campoi ≠ Id_campoj` Para toda `i,j= 1,...,n` donde `i ≠ j`
- `Tipoi` es el tipo de campo `i`

Ejemplos

- FECHA un registro formado por tres campos numéricos.
- FECHA = REGISTRO
- Día: 1..31
- Mes: 1..12
- Año: 0..2100
- {fin}

Ejemplo DOMICILIO

- DOMICILIO = REGISTRO
- Calle: cadena_de_caracteres
- Número: entero
- Ciudad: cadena_de_caracteres
- País: cadena_de_caracteres
- {fin}

Ejemplo CLIENTE

- CLIENTE= REGISTRO
- Nombre: cadena_de_caracteres
- Teléfono:
cadena_de_caracteres
- Saldo:real
- Moroso: booleano
- {fin}

Acceso a los campos de un registro

- Como un registro es un TDE no se puede tener acceso directamente, si no se debe especificar el elemento –campo que nos interesa.
- La sintaxis de la mayoría de lenguajes de POO
- `variable_registro.ide_campo`
- donde: `variable_registro` es una variable del tipo `registro`
- `id_campo` es el identificador del campo deseado

Ejemplos

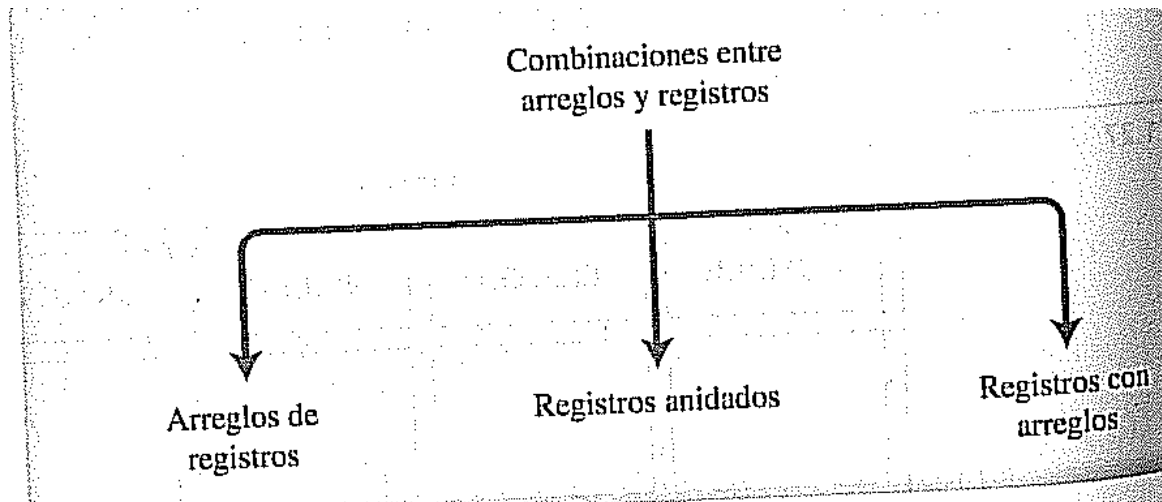
- De acuerdo con los ejemplos anteriores;
- Para leer los tres campos de una variable F de tipo FECHA
- Leer F.día, F.mes, F.año
- Para escribir los cuatro campos de una variable D de tipo DOMICILIO:
 -
 - Escribir D.calle.....D.país
- para Asignar valores a algunos campos de una variable C de tipo cliente:
 - C.saldo – C.saldo +cant
 - C.moroso – verdadero
 - C. nombre – “Juan Pérez”

1.6.3 Diferencias entre registros y arreglos

- Un arreglo puede almacenar N elementos de mismo tipo – estructura de datos homogénea-, mientras que un registro puede almacenar N elementos de diferentes tipos de datos –estructura de datos heterogénea-.
- A los componentes de un arreglo se tiene acceso por medio de índices que indican la posición del elemento correspondientes en el arreglo, mientras que los componentes de un registro, los campos, se tiene acceso por medio de su nombre, que es único

Combinaciones entre arreglos y registros

- Los registros tienen varios campos. C/u de ellos puede ser de cualquier tipo de datos, simples o estructurados. Sin embargo el nivel de más debajo de un TDE siempre debe ser un TDS.
- De acuerdo con esta condición, se infiere que un campo de un registro puede ser otro registro o bien un arreglo. Por otro lado los componentes de un arreglo pueden ser registros.

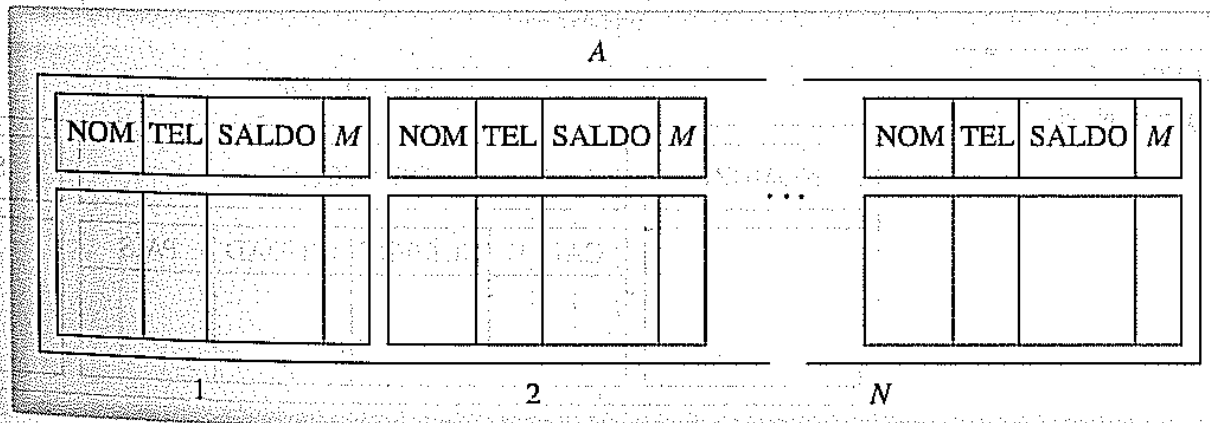


Arreglos de Registros

- ❑ En este caso, c/elemento del arreglo es un registro. Todos los componentes del arreglo tienen que ser del mismo tipo de registro, ya es un arreglo.
- ❑ Ejemplo
- ❑ Una empresa registra c/u de sus clientes con los siguientes datos:
 - ❑ CLIENTE
 - ❑ Nombre (cadena de caracteres)
 - ❑ Teléfono (cadena de caracteres)
 - ❑ Saldo (real)
 - ❑ Moroso (booleano)

- Si la empresa tiene N clientes necesitará un arreglo de N elementos, en el cual c/u de los componentes es un registro.
- $A = \text{ARREGLO}[1..100] \text{ DE CLIENTE}$
- Entonces si se quiere leer el arreglo A ;
- Repetir con I desde 1 hasta N
 - Leer $A[I].\text{nombre}$
 - Leer $A[I].\text{teléfono}$
 - Leer $A[I].\text{saldo}$
 - Leer $A[I].\text{moroso}$
- Lo mismo para las otras operaciones con arreglos

FIGURA 1.28
Arreglo de registros.



Registros Anidados

- En los registros anidados, al menos un registro es del tipo registro.
- Ejemplo 1.16 Empresa que registra sus acreedores;
- Nombre (cadena de caracteres)
- Dirección:
 - Calle: (cadena de caracteres)
 - Número: entero
 - Ciudad: (cadena de caracteres)
- País: (cadena de caracteres)
- Saldo (real)
- En este caso se declara el registro Dirección.
- Ahora se define el registro para acreedores..
- **ACREEDOR = REGISTRO**
 - nombre: cadena_de_caracteres
 - dirección: DOMICILIO
 - SALDO: REAL
- {fin}

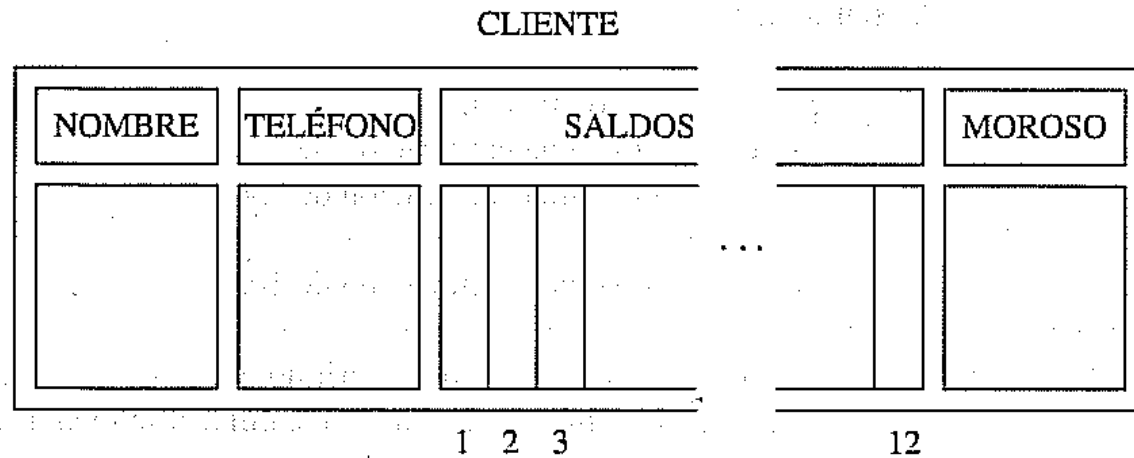
ACREEDOR

NOMBRE	DIRECCIÓN				SALDO
	CALLE	NÚMERO	CIUDAD	PAÍS	

- Para tener acceso a los datos;
- Sintaxis:
- `variable_registro.id_campo1.id_campon`
- Donde: `variable_registro` es la variable tipo registro
- `id_campo1` es el identificador de un campo, que es registro
- `id_campon` representa un identificador de un campo
- Para tener acceso a los campos;
- Variable AC de ACREEDOR
- `AC.nombre`
- `AC.dirección.calle`
- `AC.dirección.ciudad`
- `Ac.dirección.país`
- `Ac.saldo`

[Registros con Arreglos]

- Los registros con arreglos tienen, por lo menos, un campo que es de tipo arreglo.
- Ejemplo. 1.17 Registro de clientes con los sig datos;
- Nombre (cadena de caracteres)
- Teléfono (cadena de caracteres)
- Saldo mensual del último año (arreglo de reales)
- Moroso (booleano)
- Sintaxis
- CLIENTE = REGISTRO
- nombre: cadena_de_caracteres
- teléfono: cadena_de_caracteres
- saldos: ARREGLO [1..12] DE reales
- moroso:booleano
- {fin}



- Hacer referencia a los campos del arreglo;
- Variable_registro.id_campo[índice]
- Para acceder a los campos de la variable CLI de tipo CLIENTE ;
- CLI.nombre
- CLI.TELÉFONO
- Repetir con I des 1 hasta 12
- CLI.saldos [I]
- CLI.moroso

Arreglos Paralelos

- Por arreglos paralelos se entiende dos o más arreglos cuyos elementos se corresponden. En otras palabras, los componentes ocupan la misma posición en diferentes arreglos tienen una estrecha relación semántica.
- Ejemplo: Nombre de los alumnos con su calificación en el ECAL.
- Promedio, más altos del promedio o mas bajo del promedio, etc.

USO DE AREGLOS PARALELOS

- Si se utilizan arreglos paralelos se requieren dos arreglos unidimensionales tales que;
- `NOMBRES[I]` le corresponda `CALIFICACIÓN[I]`

NOMBRES		CALIFICACIONES	
1	López	1	9.5
2	Martínez	2	5.8
3	Torres	3	7.4
	⋮		⋮
30	Viasa	30	10.0

Algoritmo

Arreglos_paralelos

{Este algoritmo calcula el promedio del grupo e imprime el nombre de los alumnos con calificación menor al promedio}

{NOMBRE y CALIFICACIÓN son variables de tipo arreglo. I es una variable de tipo entero. PROM y AC son variables de tipo real}

1. Hacer $AC \leftarrow 0$

2. Repetir con I desde 1 hasta 30

 Leer NOMBRE[I] y CALIFICACIÓN[I]

 Hacer $AC \leftarrow AC + CALIFICACIÓN[I]$

3. {Fin del ciclo del paso 2}

 {Se calcula el promedio del grupo}

4. Hacer $PROM \leftarrow AC/30$

5. Escribir "El promedio del grupo es": PROM

 {Búsqueda e impresión de los nombres de los alumnos con calificación inferior al promedio}

6. Repetir con I desde 1 hasta 30

 6.1 Si (CALIFICACIÓN[I] < PROM) entonces

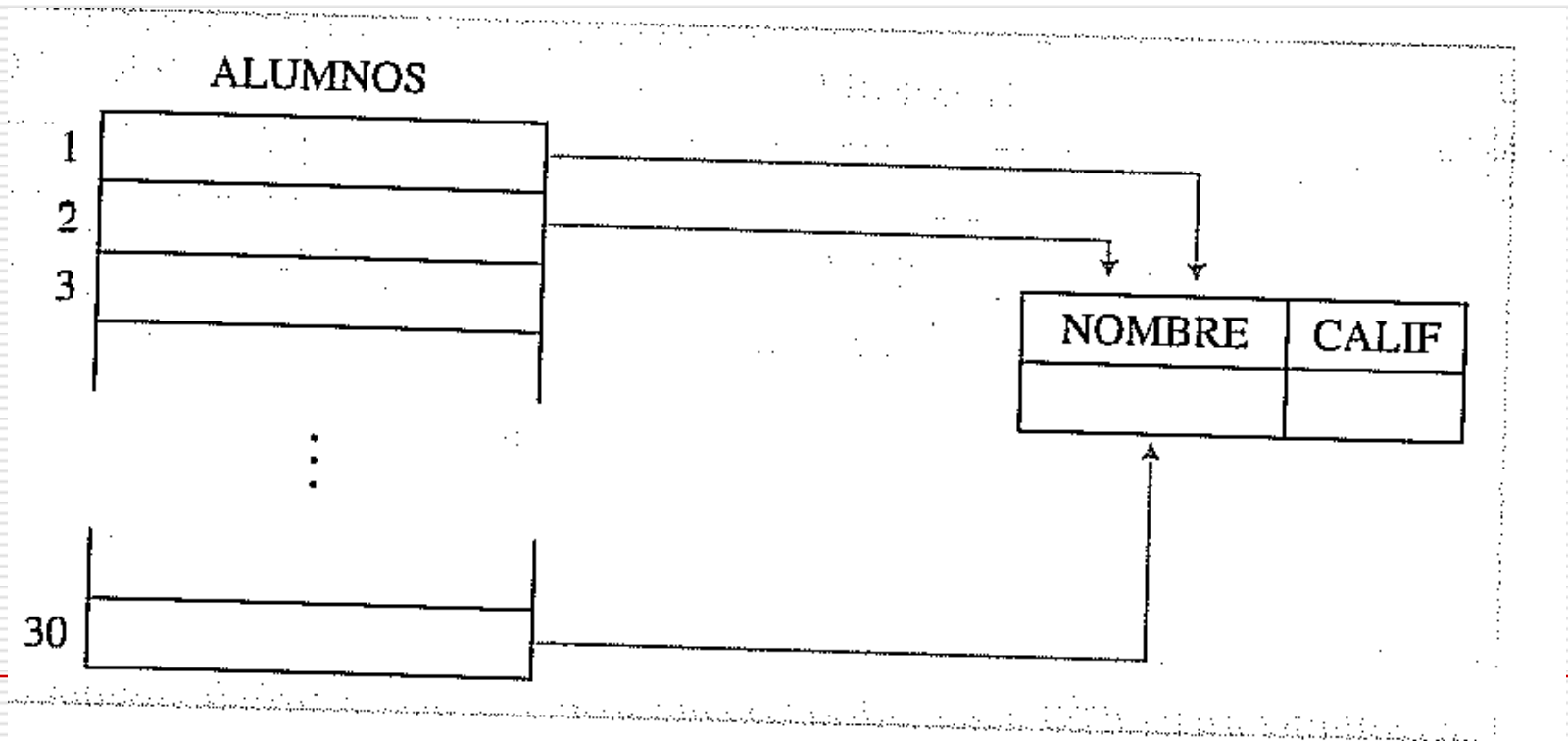
 Escribir NOMBRE[I]

 6.2 {Fin del condicional del paso 6.1}

7. {Fin del ciclo del paso 6}

USO DE AREGLOS DE RESGISTROS

- ❑ Otra solución sería el uso de un arreglo de registros.
- ❑ En este caso cada componente ALUMNO es un registro que tiene 2 campos NOMBRE Y CALIF



Se hace referencia:
ALUMNOS[I].NOMBRE
ALUMNOS[I].CALIF

□ ALGORITMO

Arreglo_de_registros

{Este algoritmo calcula el promedio del grupo e imprime el nombre de los alumnos con calificación menor al promedio}

{ALUMNOS es un arreglo de registros. *I* es una variable de tipo entero. *AC* y *PROM* son variables de tipo real}

1. Hacer $AC \leftarrow 0$
2. *Repetir* con *I* desde 1 hasta 30
 - Leer ALUMNOS[*I*].NOMBRE y ALUMNOS[*I*].CALIF
 - Hacer $AC \leftarrow AC + ALUMNOS[*I*].CALIF$
3. {Fin del ciclo del paso 2}
4. Hacer $PROM \leftarrow AC/30$
5. Escribir "El promedio del grupo es": *PROM*

{Búsqueda e impresión de los alumnos con calificación inferior al promedio}
6. *Repetir* con *I* desde 1 hasta 30
 - 6.1 Si (ALUMNOS[*I*].CALIF < *PROM*) entonces
Escribir ALUMNOS[*I*].NOMBRE
 - 6.2 {Fin del condicional del paso 6.1}
7. {Fin del ciclo del paso 6}

- TAREA
- VIERNES 25 DE AGOSTO
 - 21,24,26, 30 y 35