

Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - afeireira@exa.unrc.edu.ar

Departamento de Computación

Facultad de Cs. Exactas, Fco-Qcas y Naturales

Universidad Nacional de Río Cuarto

Teoría 10

Tipos de Datos Estructurados: Arreglos de registros, Conjuntos, Registros variantes o uniones



2017 Lic. Ariel Ferreira Szpiniak 1

Noticias Día del programador

Instituido en conmemoración al día en que Kilburn y Williams lograron hacer correr el primer programa sobre una arquitectura de von Neuman. Fue el 21 de junio de 1948.

El primer programa fue escrito por Tom Kilburn. Era un programa para encontrar el factor propio más alto de cualquier número a.

Alrededor 130.000 números fueron probados, que tomaron cerca de 2.1 millones de instrucciones

Se ha perdido el programa original, pero Tom Kilburn y Geoff Tootill reconstruyeron el primer programa.

1947/48 - Kilburn Highest Factor Routine (amended)

Instruction	C	Z	N	Y	Acc	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
245 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
246 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
247 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
248 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
249 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
250 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
251 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
252 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
253 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
254 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
255 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
256 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
257 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
258 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
259 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
260 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
261 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
262 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
263 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
264 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
265 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
266 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
267 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
268 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
269 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
270 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
271 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
272 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
273 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
274 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
275 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
276 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
277 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
278 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
279 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
280 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
281 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
282 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
283 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
284 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
285 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
286 C	-	0	-	-	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Tipos de datos

Simple

Estándar: entero, real, carácter, lógico

Definidos por el programador: subrango, enumerado

Estructurados

Estáticos: arreglos, registros, conjuntos, cadenas

Dinámicos: listas (pilas/colas), listas enlazadas, árboles, grafos

Las **estructuras estáticas** son aquellas en las que el tamaño de memoria ocupado se define antes de que el programa se ejecute y no puede modificarse durante la ejecución.

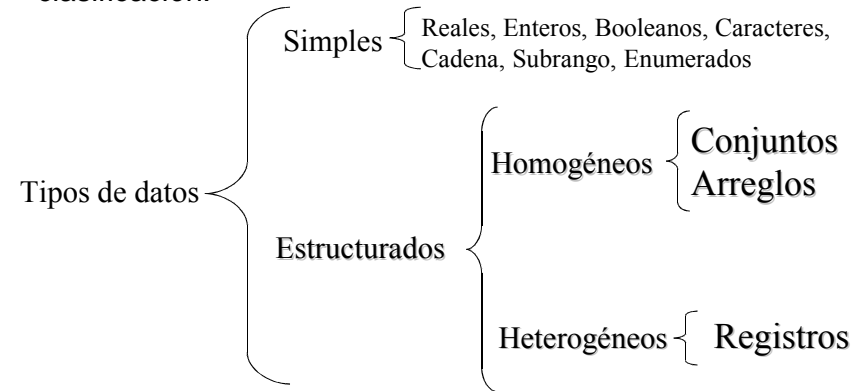
Las **estructuras dinámicas** son aquellas en las que no se debe definir previamente el tamaño de memoria.

Los **datos simples** tienen en común que cada variable representa un elemento, mientras que en los **estructurados** un identificador puede representar múltiples datos individuales, pudiendo cada uno de estos ser referenciados independientemente.

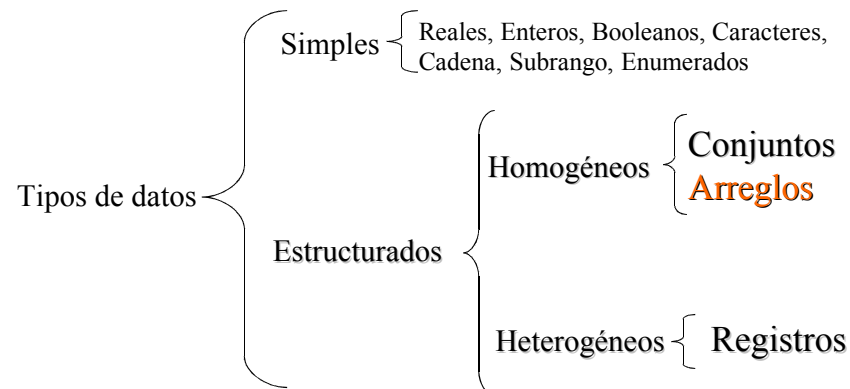


Tipos de datos

A continuación nos centraremos en el estudio de los *tipos de datos estructurados*. Para ello adoptaremos la primer clasificación:



Tipos de datos Arreglos



Arreglos Retomando....

Definición: un arreglo es una colección **finita, homogénea y ordenada** de elementos.

ordenada de elementos.

	1	2	3	4	5	6	7
tempMin	5	6	0	-3	3	8	4

		Columnas			
	puntaje	1	2	3	4
Filas	1	4	6	-2	1
	2	7	8	-4	0
	3	9	6	7	3

Tiene dos partes importantes: las **componentes** y los **índices**.

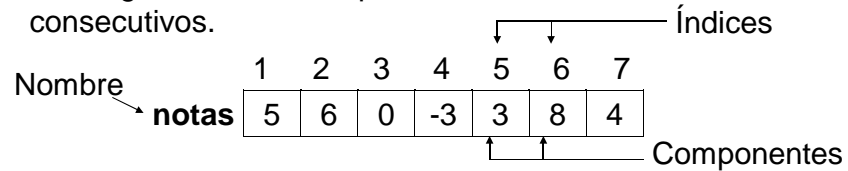
Los **componentes** hacen referencia a los elementos y los **índices** a la posición donde se encuentra el elemento.



Arreglos Unidimensionales

La estructura más simple es el arreglo de una dimensión, también llamado *vector*.

El arreglo está formado por una sucesión de elementos consecutivos.



Para referirse a una componente de un vector se utilizará el nombre y un *índice*.

El índice se encierra entre corchetes ([índice]).

Ejemplo: **notas[5]**



Arreglos Bidimensionales

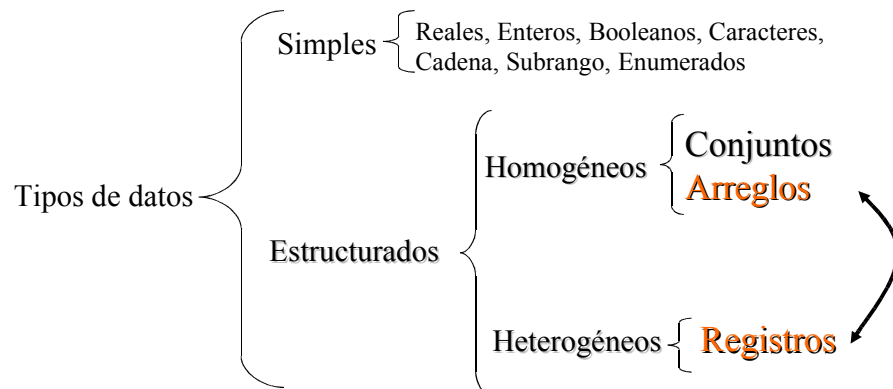
Los arreglos de dos dimensiones, también llamados *matrices*, se utilizan para representar tablas de valores.

Se requieren dos índices, uno para las **filas** y otro para las **columnas**.

		Columnas			
		1	2	3	4
Filas	1	4	6	-2	1
	2	7	8	-4	0
	3	9	6	7	3



Tipos de datos Arreglos y Registros



Arreglos de registros

Aunque los registros pueden ser útiles, muchas aplicaciones requieren de una colección de registros.

- Por ejemplo, se podría necesitar almacenar la información de todos los empleados de una empresa. Para evitar definir una variable para cada empleado, simplemente definimos un arreglo cuyas componentes son registros de tipo empleado.
- Otro caso sería por ejemplo si se desea almacenar la cantidad de milímetros llovidos en Río Cuarto en cada uno de los meses del año, la temperatura máxima registrada en cada mes, la temperatura mínima. En este caso no sería necesario definir 3 arreglos por separado sino que podría definirse:
 - Un registro de tres campos donde cada campo sea un arreglo.
 - O, lo más indicado, un solo **arreglo de registros** donde cada registro contenga 3 campos (lluvia, tempMax y tempMin).



Arreglos de registros

Ejemplo I

Supongamos que se desean guardar los datos de hasta 20 empleados (nombre, teléfono, dirección y edad).

- ¿Cómo definiríamos el tipo empleado?
- ¿Cómo definiríamos el arreglo de empleados?
- ¿Cómo realizaríamos la carga de los 15 registros del arreglo de empleados?
- ¿Cómo traduciríamos todo eso a Pascal?



Arreglos de registros - Ejemplo I

Algoritmo CargaEmpleados

Léxico

```
Max = 20
TEmpleado = <nombre ∈ Cadena, telefono ∈ Cadena,
             direccion ∈ Cadena, edad ∈ Entero>
TArregloEmpleados = arreglo[1..Max] de TEmpleado
TData = <a ∈ TArregloEmpleados, cant ∈ [0..Max+1]>
i ∈ Z
datosEmpleados ∈ TData
```

Inicio

```
datosEmpleados.cant ← 15
para i desde 1 hasta datosEmpleados.cant paso 1 hacer
    Escribir('Ingrese Nombre')
    Leer(datosEmpleados.a[i].nombre)
    Escribir('Ingrese Telefono: ')
    Leer(datosEmpleados.a[i].telefono)
    Escribir('Ingrese Dirección: ')
    Leer(datosEmpleados.a[i].direccion)
    Escribir('Ingrese Edad: ')
    Leer(datosEmpleados.a[i].edad)
```

fpara

```
Escribir('Fin de la carga, muchas gracias!')
```

Fin



Arreglos de registros - Ejemplo I

```
PROGRAM CargaEmpleados;
CONST Max = 20;
TYPE
    TEmpleado= RECORD
        nombre: STRING[20];
        telefono: STRING[10];
        direccion: STRING[20];
        edad: INTEGER;
    END;
    TArregloEmpleados = ARRAY [1..Max] OF TEmpleado;
    TData = RECORD a: TArregloEmpleados; cant: [0..Max+1]; END;
VAR
    datosEmpleados: TData;
    i: INTEGER;
BEGIN
    datosEmpleados.cant := 1;
    FOR i := 1 TO datosEmpleados.cant DO BEGIN
        WRITE(' Ingrese Nombre: '); READLN( datosEmpleados.a[i].nombre );
        WRITE(' Ingrese Telefono: '); READLN( datosEmpleados.a[i].telefono);
        WRITE(' Ingrese Dirección: '); READLN( datosEmpleados.a[i].direccion );
        WRITE(' Ingrese Edad: '); READLN( datosEmpleados.a[i].edad )
    END;
    WRITELN(' Fin de la carga, muchas gracias! ')
END.
```



Arreglos de registros - Ejemplo I

Una vez que el arreglo (cuyas componentes son registros de tipo empleado) está cargado, podríamos consultar sus valores, modificarlos, etc.

Esta manipulación se realiza de la misma manera que para arreglos cuya componentes son tipos simples, teniendo en cuenta que cada componente es un registro y por lo tanto tengo que acceder a él usando el nombre de cada campo.

Ejemplo: Supongamos que luego de la carga de los 15 empleados deseamos visualizar por pantalla el nombre y la edad de todos ellos. Agregaremos el código necesario al final del anterior...



Arreglos de registros - Ejemplo I

Algoritmo CargaEmpleados

Léxico

```
Max = 20
TEmpleado = <nombre ∈ Cadena, telefono ∈ Cadena,
             direccion ∈ Cadena, edad ∈ Entero>
TArregloEmpleados = arreglo[1..Max] de TEmpleado
TData = <a ∈ TArregloEmpleados, cant ∈ [0..Max+1]>
i ∈ Z
datosEmpleados ∈ TData
```

Inicio

```
datosEmpleados.cant ← 15
para i desde 1 hasta datosEmpleados.cant paso 1 hacer
    Escribir('Ingrese Nombre')
    Leer(datosEmpleados.a[i].nombre)
    Escribir('Ingrese Telefono: ')
    Leer(datosEmpleados.a[i].telefono)
    Escribir('Ingrese Dirección: ')
    Leer(datosEmpleados.a[i].direccion)
    Escribir('Ingrese Edad: ')
    Leer(datosEmpleados.a[i].edad)
fpara
Escribir('Fin de la carga, muchas gracias!')
{continúa en la diapositiva siguiente...}
```



2017 Lic. Ariel Ferreira Szpiniak 17

Arreglos de registros - Ejemplo I

{viene de la diapositiva anterior}

```
para i desde 1 hasta datosEmpleados.cant paso 1 hacer
    Escribir('El Nombre del empleado ',i , 'es: ')
    Escribir(datosEmpleados.a[i].nombre)
    Escribir('La edad del empleado ',i , 'es: ')
    Escribir(datosEmpleados.a[i].edad)
fpara
Escribir('Fin de la visualización!')
```

Fin



2017 Lic. Ariel Ferreira Szpiniak 18

Arreglos de registros - Ejemplo I

```
PPROGRAM CargaEmpleados;
CONST Max = 20;
TYPE
    TEmpleado= RECORD
        nombre: STRING[20];
        telefono: STRING[10];
        direccion: STRING[20];
        edad: INTEGER;
    END;
    TArregloEmpleados = ARRAY [1..Max] OF TEmpleado;
    TData = RECORD a: TArregloEmpleados; cant: [0..Max+1]; END;
VAR
    datosEmpleados: TData;
    i: INTEGER;
BEGIN
    datosEmpleados.cant := 15;
    FOR i := 1 TO datosEmpleados.cant DO BEGIN
        WRITE(' Ingrese Nombre: '); READLN( datosEmpleados.a[i].nombre );
        WRITE(' Ingrese Telefono: '); READLN( datosEmpleados.a[i].telefono);
        WRITE(' Ingrese Dirección: '); READLN( datosEmpleados.a[i].direccion );
        WRITE(' Ingrese Edad: '); READLN( datosEmpleados.a[i].edad );
    END;
    Writeln(' Fin de la carga, muchas gracias! ')
    {continúa en la dispositiva siguiente...}
```



2017 Lic. Ariel Ferreira Szpiniak 19

Arreglos de registros - Ejemplo I

{viene de la diapositiva anterior}

```
Writeln(' Fin de la carga, muchas gracias! ');
FOR i := 1 TO datosEmpleados.cant DO BEGIN
    WRITE(' El Nombre del empleado ',i , 'es: ');
    Writeln( datosEmpleados.a[i].nombre );
    WRITE(' La edad del empleado ',i , 'es: ');
    Writeln( datosEmpleados.a[i].edad );
END;
Writeln(' Fin de la visualización!')
```

END.



2017 Lic. Ariel Ferreira Szpiniak 20

Arreglos de registros - Ejemplo I

Algoritmo CargaEmpleados

Léxico

Max = 20

TEmpleado = <nombre ∈ Cadena, telefono ∈ Cadena,
direccion ∈ Cadena, edad ∈ Entero>

TArregloEmpleados = arreglo[1..Max] de TEmpleado

TData = <a ∈ TArregloEmpleados, cant ∈ [0..Max+1]>

i ∈ Z

datosEmpleados ∈ TData

Inicio

Escribir('Ingrese la cantidad de empleados de la Empresa')

Leer(datosEmpleados.cant)

para i desde 1 hasta datosEmpleados.cant paso 1 hacer

Escribir('Ingrese Nombre')

Leer(datosEmpleados.a[i].nombre)

Escribir('Ingrese Telefono: ')

Leer(datosEmpleados.a[i].telefono)

Escribir('Ingrese Dirección: ')

Leer(datosEmpleados.a[i].direccion)

Escribir('Ingrese Edad: ')

Leer(datosEmpleados.a[i].edad)

finpara

Escribir('Fin de la carga, muchas gracias!')

Fin



2017 Lic. Ariel Ferreira Szpiniak 21

Arreglos de registros Ejemplo II

Desarrollar un algoritmo que solicite al usuario la cantidad de milímetros llovidos en Río Cuarto en cada uno de los meses del año, la temperatura máxima registrada en cada mes y la temperatura mínima. El algoritmo debe almacenar los datos en un arreglo, luego calcular el total de precipitación anual, el mes en que aconteció la temperatura máxima y el mes en que aconteció la mínima. Finalmente debe informarlo por pantalla.



2017 Lic. Ariel Ferreira Szpiniak 22

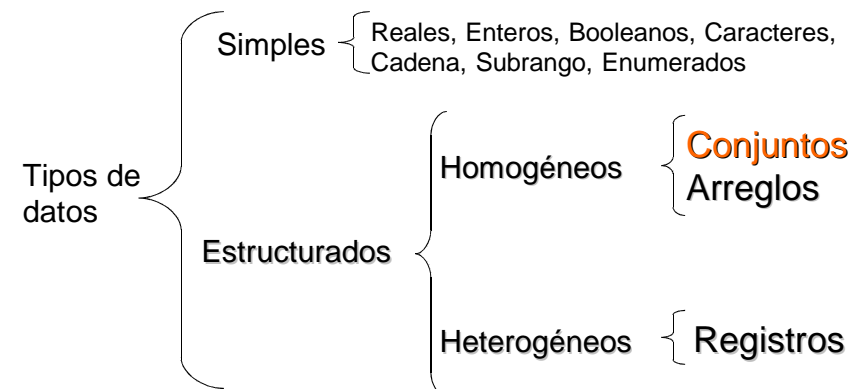
Arreglos de registros Ejercicio III

1. Rehacer el Ejemplo I definiendo dos acciones con parámetros, una para la carga, y otra para listar el nombre y edad de cada empleado. Pasar el arreglo como parámetro.
2. Agregar una nueva acción que permita al usuario listar los **n** primeros empleados, con **n** a elección.
3. Agregar al algoritmo principal un menú que permita la usuario elegir:
 - a) Cargar 15 y luego Mostrar nombre y edad de los 15.
 - b) Cargar entre 1 y 20 y luego listar todos lo empleados recientemente cargados.
4. Traducir a Pascal



2017 Lic. Ariel Ferreira Szpiniak 23

Tipos de datos Conjuntos



2017 Lic. Ariel Ferreira Szpiniak 24

Conjuntos

Es un tipo de dato estructurado en el que las variables pueden almacenar varios valores de un tipo simple, al cual se le denomina tipo base. La declaración de un tipo conjunto se hace:

Notación algorítmica

nombre \in CONJUNTO de tipobase

Ejemplo:

```
TConjuCar = CONJUNTO de
                        Caracter
conjuCar  $\in$  TConjuCar
```

Pascal

TYPE nombre: **SET OF** tipo;

Ejemplo:

```
TYPE TConjuChar = SET OF Char;
VAR conjuChar: TConjuChar;
```

- El tipo Conjunto es muy dependiente del lenguaje utilizado.
- Cada lenguaje tiene su propia forma de representarlo.
- No es muy usado.



Conjuntos

Operadores con conjuntos:

+ : Unión de conjuntos.

- : Diferencia de conjuntos.

* : Intersección de conjuntos.

EN / IN: Pertenencia en Pseudocódigo y en Pascal.

>= : Un conjunto incluye a otro conjunto.

<= : Un conjunto es incluido en otro conjunto.

Modo de Uso en Pascal

conjuChar := []; {se asigna conjunto vacío, es obligatorio}

conjuChar := conjuChar+['A']; {UNION}

IF ('A' IN conjuChar) THEN WRITELN('A está en el conjunto') {PERTENENCIA}



Conjuntos

Ejemplo

Algoritmo Colores

Léxico

```
TColorPrimario = (Rojo, Azul, Amarillo)
TConjuntoColor = CONJUNTO de TColorPrimario
color  $\in$  TConjuntoColor
```

Inicio

```
color  $\leftarrow$  []
color  $\leftarrow$  [Rojo]
color  $\leftarrow$  color + [Amarillo]
color  $\leftarrow$  color * (color - [Rojo])
si (Azul EN color)
```

entonces

```
    Escribir('Azul pertenece al conjunto')
```

sino

```
    Escribir('Azul no pertenece al conjunto')
```

fsi

Fin



Conjuntos

Ejemplo en Pascal

```
PROGRAM Colores;
```

```
TYPE
```

```
    TColorPrimario = (Rojo, Azul, Amarillo);
    TConjuntoColor = SET OF TColorPrimario;
```

```
VAR
```

```
    color: TConjuntoColor;
```

```
BEGIN
```

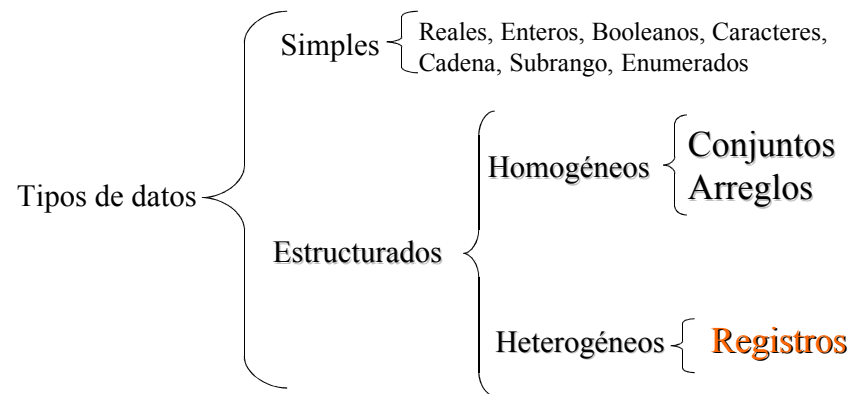
```
    color := [];
    color := [Rojo];
    color := color + [Amarillo];
    color := color * (color - [Rojo]);
    IF (Azul IN color)
    THEN WRITELN('Azul pertenece al conjunto')
    ELSE WRITELN('Azul no pertenece al conjunto')
```

```
END.
```



Tipos de datos

Registros

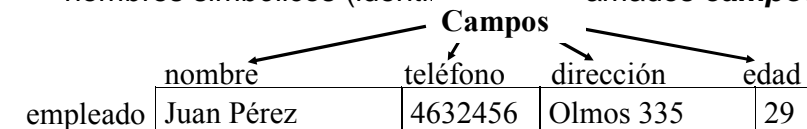


Registros

Un registro es un mecanismo mediante el cual se pueden agrupar elementos de varios tipos. De esta manera es posible representar o modelar entidades u objetos del mundo real, como por ejemplo un **empleado**.

Los registros están formados por **componentes**.

1. Los **componentes** de los registros pueden ser heterogéneos, por ejemplo, tipos de datos mixtos (strings, enteros, reales...).
2. Los **componentes** pueden ser simples o estructurados.
3. Los **componentes** de los registros se nombran con nombres simbólicos (identificadores) llamados **campos**.

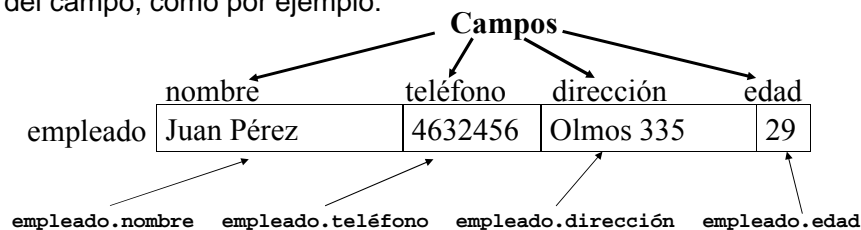


Registros

Selección de componentes

En Notación Algorítmica y en Pascal la **selección de una componente** se realiza haciendo referencia al nombre del **campo**.

Esto se logra colocando el nombre del registro punto el nombre del campo, como por ejemplo:



De esta manera se puede acceder al contenido de cada campo (leer) así como también se puede colocar información dentro de cada uno (escribir).



Registros variantes o Uniones

Con frecuencia es necesario caracterizar entidades que comparten algunos atributos y difieren en otros.

Los **registros variantes**, también conocidos como **uniones**, permiten representar adecuadamente esta situación.

- Un registro variante está compuesto por una parte **fija** y una parte **variable**.
- La parte **fija** de un registro variante consiste de aquellos campos que son *comunes* a todos los registros de ese tipo.
- La parte **variable** declara los campos que pueden existir dependiendo del valor de un cierto campo, llamado **campo etiqueta**.



Ejemplo

Supongamos tenemos estudiantes de grado y posgrado, donde tienen en común el nombre, documento y dirección pero difieren en que los estudiantes de grado tienen el año de ingreso a la carrera y los de posgrado los nombres del director y codirector de tesis.

En Pseudocódigo se podría definir de la siguiente manera:

```
TEstudiante = < nombre ∈ Cadena, doc ∈ Cadena,
               direccion ∈ Cadena,
               status ∈ (grado, posgr):
                   grado ∈ (anioingreso ∈ Z)
                   posgr ∈ (director ∈ Cadena,
                           codirector ∈ Cadena)
```

>

etiqueta



Ejemplo (cont.)

Un Algoritmo que realice la carga de un estudiante.

Algoritmo CargaEstudiante

Léxico

```
TEstudiante = < nombre ∈ Cadena, doc ∈ Cadena, direccion ∈ Cadena,
               status ∈ (grado, posgr):
                   grado ∈ (anioIngreso ∈ Z)
                   posgr ∈ (director ∈ Cadena, codirector ∈ Cadena)
```

>

```
est ∈ TEstudiante
claseEst ∈ Caracter
```

Inicio

```
Escribir(' Ingrese Nombre del estudiante: ')
Leer(est.nombre )
Escribir(' Ingrese Documento: ')
Leer(est.doc)
Escribir(' Ingrese Dirección: ')
Leer( est.direccion )
Escribir(' Ingrese "G" si es de grado o "P" si es de posgrado: ')
Leer(claseEst )
{continúa en la dispositiva siguiente...}
```



Ejemplo (cont.)

{viene de la dispositiva anterior}

```
si claseEst="G"
entonces est.status ← grado
sino est.status ← posgr
fsi

segun

    est.status = grado: Escribir('Ingrese el año de ingreso del estudiante')
                        Leer(est.anioIngreso)

    est.status = posgr: Escribir('Ingrese el nombre del director')
                        Leer(est.director)
                        Escribir('Ingrese el nombre del codirector')
                        Leer(est.codirector)
```

fsegun

Fin



Ejemplo

Supongamos tenemos estudiantes de grado y posgrado, donde tienen en común el nombre, documento y dirección pero difieren en que los estudiantes de grado tienen el año de ingreso a la carrera y los de posgrado los nombres del director y codirector de tesis.

En Pascal se podría definir de la siguiente manera:

TYPE TEstudiante = RECORD

```
    nombre: STRING[40];
    doc: STRING[20];
    direccion: STRING[30];
    CASE status: (grado, posgr) OF
        grado: (anioIngreso: INTEGER);
        posgr: (director: STRING[40];
               codirector: STRING[40];)
```

END;

etiqueta



Ejemplo (cont.)

Un programa que realice la carga de un estudiante.
En Pascal se podría definir de la siguiente manera:

```
PROGRAM CargaEstudiante;
TYPE TEstudiante = RECORD
    nombre: STRING[40];
    doc: STRING[20];
    direccion: STRING[30];
    CASE status: (grado, posgr) OF
        grado: (anioIngreso: INTEGER);
        posgr: (director: STRING[40]; codirector: STRING[40]);
    END;
VAR
    est: TEstudiante;
    claseEst: CHAR;
BEGIN
    WRITE(' Ingrese Nombre del estudiante: ');
    READLN(est.nombre);
    WRITE(' Ingrese Documento: ');
    READLN(est.doc);
    {continúa en la dispositiva siguiente}
```



2017 Lic. Ariel Ferreira Szpiniak 37

Ejemplo (cont.)

```
{viene de la diapositiva anterior}
WRITE(' Ingrese Dirección: ');
READLN( est.direccion );
WRITE(' Ingrese "G" si es de grado o "P" si es de posgrado: ');
READLN(claseEst);
IF (claseEst='G')
THEN est.status:=grado
ELSE est.status:= posgr;
CASE est.status OF
    grado: BEGIN
        WRITELN(' Ingrese el año de ingreso del estudiante');
        READLN(est.anioIngreso);
    END;
    posgr: BEGIN
        WRITELN(' Ingrese el nombre del director ');
        READLN(est.director);
        WRITELN(' Ingrese el nombre del codirector ');
        READLN(est.codirector);
    END;
END;
END.
```



2017 Lic. Ariel Ferreira Szpiniak 38

Ejemplo II

Un **vuelo** de **avión** que posee **código** y **destino**. Pero si el vuelo es **nacional** se sabe la cantidad de **horas** que demora (entre 1 y 12, sin fraccionar) y si es **internacional** se sabe las **escalas** que realiza (tres como máximo) además de si es un avión de **porte** chico, mediano o grande.

En Pascal se podría definir de la siguiente manera:

```
TYPE THora = 1..12;
TEscalas = ARRAY [1..3] OF STRING[20];
Tavion = (Chico, Mediano, Grande);
TVuelo = RECORD
    codigo: STRING[5];
    destino: STRING[20];
    CASE tipo: (nac,internac) OF
        nac: (tiempovuelo: THora);
        internac: (esc: TEscalas;
            porte: Tavion);
    END;
```



2017 Lic. Ariel Ferreira Szpiniak 39

Ejemplo II (cont.)

Un programa que muestre un vuelo ya cargado en un registro de tipo Vuelo. En Pascal se podría definir de la siguiente manera:

```
PROGRAM MostrarVuelo;
TYPE TVuelo = RECORD .... END;
VAR vueloActual: TVuelo; i: INTEGER; claseEst: CHAR;
BEGIN
    ....
    WRITELN(' El código del vuelo es: ', vueloActual.codigo);
    WRITELN(' El destino del vuelo es: ', vueloActual.destino);
    CASE vueloActual.tipo OF
        nac: WRITELN(' El tiempo de vuelo es: ', vueloActual.tiempovuelo);
        internac: BEGIN
            FOR i:=1 TO 3 DO BEGIN
                WRITELN(' La Escala ', i, ' es: ', vueloActual.esc[i]);
            END;
            CASE vueloActual.porte OF
                Chico: WRITELN(' El Avión es de tipo Chico ');
                Mediano: WRITELN(' El Avión es de tipo Mediano ');
                Grande: WRITELN(' El Avión es de tipo Grande ');
            END;
        END;
    END;
```



2017 Lic. Ariel Ferreira Szpiniak 40

Ejemplo II (cont.)

Una Concesionaria de autos necesita al programa para cargar los datos de los automóviles en venta (200 como máximo) y luego mostrarlos según ciertos criterios. De cada auto se conoce la marca, modelo, año, km, y si es un auto base o full. De los autos full se registra si tiene AA o climatizador, frenos ABS, airbag, y cantidad de airbag.

Desarrollar un algoritmo que permita almacenar hasta 200 autos y luego mostrar por pantalla todos los que cumplen con ciertos criterios a elección del usuario:

- por cantidad de km,
- los base,
- los full,
- los que tienen AA,
- los que tienen airbag.



2017 Lic. Ariel Ferreira Szpiniak 41

Bibliografía

- Scholl, P. y J.-P. Peyrin, “Esquemas Algorítmicos Fundamentales: Secuencias e iteración”, Barcelona, Ed. Masson, 1991.
- Lucas, M., J.-P. Peyrin y P. Scholl, “Algorítmica y Representación de Datos. Tomo 1: Secuencia, Autómata de estados finitos”, Barcelona, Ed. Masson, 1985.
- Watt, David, “Programming Language Concepts and Paradigms“, Prentice-Hall International Series in Computer Science (1990).
- Biondi, J. y G. Clavel, “Introducción a la Programación. Tomo 1: Algorítmica y Lenguajes”, 2° ed., Barcelona: Masson, 1985.
- Clavel, G. y Biondi, J., “Introducción a la Programación. Tomo 2: Estructuras de Datos”, 2° ed., Barcelona: Masson, 1985.
- De Guisti, A. “Algoritmos, datos y programas. Con aplicaciones en Pascal, Delphi y Visual Da Vinci. Prentice Hall.
- Joyanes Aguilar, L., “Programación en Turbo Pascal”. Mc Graw Hill, 1993.



2017 Lic. Ariel Ferreira Szpiniak 42

Citar/Atribuir: Ferreira, Szpiniak, A. (2017). Teoría 10: Tipos de Datos Estructurados: Arreglos de registros, Conjuntos, Registros variantes o uniones. Introducción a la Algorítmica y Programación (3300). Departamento de Computación. Facultad de Cs. Exactas, Fco-Qcas y Naturales. Universidad Nacional de Río Cuarto.

Usted es libre para:

Compartir: copiar y redistribuir el material en cualquier medio o formato.

Adaptar: remezclar, transformar y crear a partir del material.

El licenciente no puede revocar estas libertades en tanto usted siga los términos de la licencia.

Bajo los siguientes términos:



Atribución: Usted debe darle crédito a esta obra de manera adecuada, proporcionando un enlace a la licencia, e indicando si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo del licenciente.



Compartir Igual: Si usted mezcla, transforma o crea nuevo material a partir de esta obra, usted podrá distribuir su contribución siempre que utilice la misma licencia que la obra original.

<https://creativecommons.org/licenses/by-sa/2.5/ar/>

