Slide 1

TEMA 10: Tipos estructurados: Registro. Array.

Índice

Slide 2

Tipos de datos compuestos

- Tipo de sus componentes (homogéneo/heterogéneo)
- Tamaño (número de componentes: fijo, variable, limitado, indeterminado)
- Organización (forma fija/variable)
- Modo de acceso a las componentes
- Persistencia en memoria (permanentes/transitorios)
- Implantación en el lenguaje (si/no/parcial)
- Estándar del lenguaje

Slide 3

	comp	tam	org	acceso	pers	imp	est
vector	hom	fijo	lin	índice	T	$\frac{1}{2}$	si
matriz	hom	fijo	mat	índices	T	$\frac{1}{2}$	si
cadena	char	var/M	lin	posic	T	$\frac{1}{2}$	no
registro	het	fijo	irr	nomb	T	$\frac{1}{2}$	si
fichero	hom	indet	lin	sec	P	$\frac{1}{2}$	si
	het	indet	lin	sec	P	$\frac{1}{2}$	si
conjunto	hom	var/M	-	$\in$	T	$\frac{1}{2}$	si
lista	hom	var	lin	sec	T	no	
pila	hom	var	lin	LIFO	T	no	
cola	hom	var	lin	FIFO	T	no	
árbol	hom	var	var	...	T	no	

Slide 4

TIPO DE DATOS VECTOR

Número fijo de componentes homogéneos, organizados linealmente, acceso por índice. Transitorio, semidefinido, *estándar*.

Valores: ($TIndice$ ordinal y $TBase$ de tamaño fijo)

$$TBase^{TIndice} = TBase \times TBase \times \ldots \times TBase$$

array [<*TIndice*>] **of** <*id-TBase*>

Operaciones:

- Acceso a cada componente por su índice (`{$R+$}` ó `-Cr`)

  <*id-var*> [<*expresión-TIndice*>]
- Para cada componente las de su tipo
- *Asignación*
- Paso como parámetro *(tipo con nombre)*
- Las que el programador defina

Slide 5

array [1..100] **of** real;

array [0..10] **of** integer;

array ['a'..'z'] **of** integer;

array [1..100] **of** boolean;

array [-10..10] **of** PosInt;

array [INF..SUP] **of** char;

array [TDiaSemana] **of** integer;

array [lun..vie] **of** integer;

---

var nota1, nota2, notaM : **array** [1..100] **of** real;

var NCalif : **array** [0..10] of Integer;

type TContador = **array** ['a'..'z'] **of** integer;

var frecuencia : TContador;

var Par : **array** ['a'..'z'] **of** TContador;

var Aprobado : **array** [1..100] **of** boolean;

var HorasT : **array** [TDiaSemana] **of** integer;

Slide 6

```
nota1[35] := 5;
for i := 1 to 100 do readln (nota1[i]); (* paso por variable *)
for i := 1 to 100 do writeln (nota1[i]); (* paso por valor *)
notaM[i] := (nota1[i] + Nota2[i]) /2;
Aprobado[i] := notaM[i]>=5;
Calif[round(notaM[i])]:= Calif[round(notaM[i])] + 1;
if not Aprobado[i] then HorasT[sab] := 5;
frecuencia['a'] := 0;
Par['i']['u'] := Par['i']['u'] +1;
if odd(Par['i']['u']) then ... (* paso por valor *)
incrementa(Par['i']['u']); (* paso por variable *)
```

Slide 7

```
program temperaturas (input, output);
const HINI = 1; HFIN = 24;
var  temp, tmax, tmin, suma: real;
     h : integer; (* hora *)
BEGIN
  h := HINI;
  write ('Temp. a las ', h, ' horas:');
  readln (temp);
  suma := temp; tmax := temp; tmin := temp;

  for h := HINI + 1 to HFIN do begin
        write ('Temp. a las ', h);
        readln (temp);
        suma := suma + temp;
        if temp > tmax then tmax := temp
        else if temp < tmin then
             tmin := temp;
      end;
  writeln ('Máx: ', tmax, ' Mín: ', tmin,
           'Med: ', suma/(HFIN-HINI+1))
END.
```

Slide 8

```
program tempYGrafico (input, output);
const
     HINI = 1; HFIN = 24;
var
     temp : array [HINI..HFIN] of real;
     tmax, tmin, suma: real;
     h: integer; (* hora *)
     i : integer; (* contador  de * *)
BEGIN
 (* Leer temperaturas *)
   for h := HINI to HFIN do
        begin
             write ('Temp. a las ', h);
             readln (temp[h]);
        end;

 (* Calcular máx, mín y suma (para media) *)
   tmax := temp[HINI];
   tmin := temp[HINI];
   suma := temp[HINI];
```

Slide 9

```
   for h := HINI + 1 to HFIN do begin
              suma := suma + temp[h];
              if temp[h] > tmax then
                   tmax := temp[h]
              else if temp[h] < tmin then
                   tmin := temp[h] ;
         end;
   writeln ('Máx: ', tmax, ' Mín: ', tmin,
             'Med: ', suma/(HFIN-HINI+1));
 (* pintar Gráfico *)
   writeln ('GRAFICO:');
   for h := HINI to HFIN do
         begin
              write (h:3);
              (* Escribir en * temp-tmin*)
              for i := 1 to
                  trunc(temp[h]-tmin)
                 do write ('*');
              writeln;
         end
END.
```

Slide 10

```
program elecciones (input, output);
const
     ELECTORES = 100;
     ELEGIBLES = 5;
     NULO = -1;
     BLANCO = 0;

var
     recuento: array [NULO..ELEGIBLES] of integer;
     elector : 1..ELECTORES;
     voto: integer;
     indice : NULO..ELEGIBLES;

Begin
     for indice := NULO to ELEGIBLES do
          recuento [indice] := 0;
```

Slide 11

```
     for elector := 1 to 100 do
          begin
              write ('Valor del voto ', elector) ;
              readln (voto);
              if (voto < 0) or (voto > ELEGIBLES) then
                  recuento[NULO] := recuento[NULO] + 1
              else
                  recuento[voto] := recuento[voto] + 1
          end;
     writeln ('Nulos: ', recuento[NULO]);
     writeln ('Blanco: ', recuento[BLANCO]);
     for indice := 1 to ELEGIBLES do
          writeln ('Candidato ', indice:2, ': ',
                    recuento[indice])
End.
```

Slide 12

```
function posMax
   (var v : tvector; idesde, ihasta : integer): integer;
(* Pre:Tvector=array[P..Q] of real; P<=idesde<=ihasta<=Q
(* devuelve la posición del primer máximo del vector v
(* entre las posiciones idesde y ihasta *)
var iprov : integer; maxprov : real; i : integer;
begin
    maxprov := v[idesde]; iprov := idesde;
    for i := idesde +1 to ihasta do
        if v[i]>maxprov then
           begin
               maxprov := v[i];
               iprov := i
           end;
    posMax := iprov
end;
```

Slide 13

```
procedure normalizaMax  (* normaliza MAL *)
   (var v: Tvector; idesde, ihasta : integer);
var imax : integer; i : integer;
begin
    imax := posMax (v, idesde, ihasta);
    for i := idesde to ihasta do v[i]:=v[i]/v[imax]
end;
```

Slide 14

Operaciones típicas

Iniciación (asignación de valor)

Recorrido

Recorrido con tratamiento selectivo

Búsqueda

Eliminación de un elemento

Inserción de un elemento

Slide 15

Recorrido; Recorrido selectivo

type TBase = . . . ; TIndice = . . . ;
TVector = **array** [TIndice] **of** TBase;
var v : TVector; i : TIndice ;

i_0 e i_f : expresiones evaluables de tipo TIndice

para i **desde** i_0 **hasta** i_f **hacer**
Tratamiento ($v[i]$)
fin para

selector : TBase × TIndice ⟶ **Boolean**

para i **desde** i_0 **hasta** i_f **hacer**
si selector(i, $v[i]$) **entonces**
Tratamiento ($v[i]$)
fin si
fin para

Slide 16

Búsqueda secuencial

obj expresión evaluable de tipo TBase

Caso simple: encontrar

```
i ← i_0
mientras v[i] ≠ obj hacer
            i ← succ(i)
fin mientras
(* v[i] = obj *)
{ encontrado en i }
```

Slide 17

Búsqueda secuencial

Caso general: buscar. El elemento *puede no estar* en el vector.

```
i ← i_0
mientras i<i_f y v[i] ≠ obj hacer
            i ← succ(i)
fin mientras
(* i = i_f o i < i_f y v[i] = obj *)
si v[i] = obj entonces
                { encontrado en i }
si no
                { no encontrado }
fin si
```

Slide 18

```
procedure SuperMax
            (var v : Tvector; idesde, ihasta: integer;
             var m: real; var pos, num: integer);
(* Entradas: v: vector; idesde, ihasta: rango de búsqueda
   Salidas : en m el valor del máximo del vector
             en pos la posición de la primera aparición
             en num el núm. de veces que se alcanza *)
var i: integer;
begin
    m := v[idesde]; pos := idesde; num := 1;
    for i := succ(idesde) to ihasta do
        if v[i]>m then begin
                  m := v[i]; pos := i; num := 1
           end
        else if v[i]=m then num := num + 1
end;
```

Slide 19

```
procedure MostrarMaximos
           (var v : TVector; idesde, ihasta: integer);
var  pos, nveces : integer; max : real;
begin
    SuperMax (v, idesde, ihasta, max, pos, nveces);
    write (pos:3);
    while nveces>1 do
       begin
          pos := encuentra (v, pos+1, max);
          write (pos:3);
          nveces := nveces-1
       end;
   writeln
end;
```

Slide 20

Tipo de datos Lista[1]

Número variable (hasta un máximo)[1] de componentes homogéneos, organizados linealmente, acceso secuencial. Transitorio, definido por el programador.

Valores: ...

Operaciones:

- ...
- Inserción de un elemento
- Eliminación de un elemento

Implantación:
vector de tamaño máximo + variable para el tamaño *efectivo*

Slide 21

```
type TNotas = array[1..100] of real;
procedure LeerNotas (var v: TNotas; var Pres: integer);
  (* leer hasta nota negativa *)
  (* Salida: Pres: n. de presentados. v: sus notas *)
  (* Sup: no se introducen más de 100 notas *)
var nota : real;
begin
   pres := 0;
   write ('Alumno', pres+1); readln (nota);
   while nota >= 0 do
      begin
          pres := pres + 1;
          v[pres] := nota;
          write ('Alumno', pres+1); readln (nota)
      end
end;
```

Slide 22

TIPO DE DATOS MATRIZ (TABLA)

Número fijo de componentes homogéneos, organizados multidimensionalmente, acceso por índices. Transitorio, semidefinido, *estándar*.

Valores: ...

$TIndice_1$, $TIndice_2$, ..., $TIndice_p$, ordinales y $TBase$ de tamaño fijo

array [<*TIndice*> $_1$ {, <*TIndice*> $_i$}] **of** <*id-TBase*>

Operaciones: Las mismas que para los vectores

<*id-var*> [<*expresión-TIndice*> $_1$ {, <*expresión-TIndice*> $_i$ }]

Slide 23

```
var dig : array ['a'..'z', 'a'..'z'] of integer;
var m : array [lun..vie, 'a'..'z', -1..3] of integer;
var b : array [1..100, -1..1, 0..4] of boolean;
...
dig [ 'a', 'u' ] := dig [ 'a', 'u' ] + 1 ;
dig [ succ('a'), pred('b') ] := 0;
readln (input, m [ jue, 's', 3 DIV 2 ] ) ;
if b[1,1,1] then ...
```

Slide 24

```
program Medias (input, output);
(* Entrada: Serie de 3 notas parciales de 100 alumnos
   Salida : Media de cada parcial (de los 100 alumnos)
            Media de las notas finales
             (la final es la media de las parciales
   Datos: se almacenan las notas en una matriz
          siendo la 4ª columna la nota final *)

const NAL = 100;
      NNOTAS = 3;
type TNotas = array [1..NAL, 1..NNOTAS+1] of real;
var  nota : TNotas;
     n: 1..NNOTAS+1;
     al : 1..NAL;
     s : real;
```

Slide 25

```
procedure LeerNotasYCalcMedias (var m: TNotas);
var al: integer;
    n : integer;
    s : real;
begin
    for al := 1 to NAL do
        begin
          writeln ('Alumno ', al); s := 0;
          for n := 1 to NNOTAS do
              begin
                write ('Nota ', n, ':'); readln (m[al,n]);
                s := s + m[al,n];
              end;
          m[al,NNOTAS+1] := s/NNOTAS
       end;
end;
```

Slide 26

```
BEGIN
    LeerNotasYCalcularMedias (nota);
    for n := 1 to NNOTAS do
       (* media del parcial n *)
        begin
            s := 0;
            for al := 1 to NAL do s := s + nota[al,n];
            writeln ('Media del parcial ', n,':', s/NAL);
        end;
    (* Media de las finales *)
    s := 0;
    for al := 1 to NAL do s := s + nota[al,NNOTAS+1];
    writeln ('Media de las finales ', n,':', s/NAL);
END.
```

Slide 27

```
program sumaMatrices (input, output);
const NF=5, NC=4;
type TMatriz = array[1..NF, 1..NC] of real;
var  a, b, c: TMAtriz;
procedure LeerMatriz (var m: TMatriz);
...
procedure EscribirMatriz (var m : TMatriz);
...
procedure SumarMat (var m1, m2, m3: TMatriz);
...
BEGIN
   LeerMatriz (a);
   LeerMatriz (b);
   SumarMat (a, b, c);
   EscribirMatriz (c)
END.
```

Slide 28

```
procedure LeerMatriz (var m: TMatriz);
(* Lee m de teclado, por filas *)
var i, j : integer;
begin
    for i := 1 to NF do
        begin
            writeln ('Fila ', i);
            for j := 1 to NC do
                begin
                   write ('  Elemento', j);
                   readln (m[i, j]);
                end
        end
end;
```

Slide 29

```
procedure EscribirMatriz (var m : TMatriz);
var i, j : integer;
begin
    for i := 1 to NF do
        begin
            (* Fila i *)
            for j := 1 to NC do
                   write (m[i,j]:7:2);
            writeln
        end
end;
```

Slide 30

```
procedure SumarMat (var m1, m2, m3: TMatriz);
var i, j : integer;
begin
    for i := 1 to NF do
        for j := 1 to NC do
               m3[i,j] := m1[i,j] + m2[i,j]
end;
```

Slide 31

```
program FuncionesConMatrices(input, output);
const NMAX = 10;
type tmatriz =
        array [1..NMAX, 1..NMAX] of real;
var a, t , c: tmatriz;
    nf, nc : integer; (* tamaño efectivo *)
BEGIN
    write ('Dimensiones de la matriz:'
           ' (Máximo 10x10) ');
    readln (nf, nc); (* Sup. correctas *)
    LeerMatriz (a, nf, nc);
    EscribirMatriz (a, nf, nc); readln;
    Trasponer (a, nf, nc, t);
    writeln ('Traspuesta: ')
    EscribirMatriz (t, nc, nf); readln;
    writeln ('Matriz por su traspuesta: ');
    Multiplicar (a, t, nf, nc, nf, c);
    EscribirMatriz (c, nf, nf); readln;
    writeln ('Traspuesta por la matriz: ');
    Multiplicar (t, a, nc, nf, nc, c);
    EscribirMatriz (c, nc, nc);
END.
```

Slide 32

```
procedure LeerMatriz (var m : Tmatriz; nf, nc : integer);
(* Tamaño real : NMAX x NMAX. Tamaño efectivo: nf x nc *)
    var i, j : integer;
begin
    writeln ('Lectura de la matriz por filas ');
    for i := 1 to nf do
        begin (* fila i *)
            writeln  ('Fila ', i, ':',
                      nc, 'elementos entre espacios');
            for j := 1 to nc do
                begin (* elemento i, j *)
                    read (m[i,j])
                end;
            writeln
        end
end;
```

Slide 33

```
procedure EscribirMatriz
             (var m : Tmatriz; nf, nc : integer);
(* Tamaño real : NMAX x NMAX. Tamaño efectivo: nf x nc *)
    var i, j : integer;
begin
    writeln ('Escritura de la matriz por filas ');
    for i := 1 to nf do
        begin (* fila i *)
            for j := 1 to nc do
                   write (m[i,j]);
            writeln
        end
end;
```

Slide 34

```
procedure Trasponer
     (var a: Tmatriz; nf, nc: integer; var t : tmatriz);
(* Tamaño real de a y t : NMAX x NMAX.
   Tamaño efectivo de a: nf x nc *)
    var i, j : integer;
begin
    for i := 1 to nf do
        for j := 1 to nc do
            t [j, i] := a[i, j]
end;
```

Slide 35

```
procedure Multiplicar
   (var a, b: tmatriz; m, n, p: integer; var c: tmatriz);
(* Tamaño real : NMAX x NMAX.
   Tamaños efectivos de a, b y c:
               m x n,  n x p y  m x p respectivamente *)
    var   i, j, k : integer; s : real;
begin
    for i := 1 to m do
        for j := 1 to p do
            begin
                s := 0;
                for k := 1 to n do
                    s := s + a[i, k]*b[k, j];
                c[i, j] := s
            end
end;
```

Slide 36

Tipo de datos Registro

Número fijo de componentes no homogéneos, no organizados, acceso por nombre. Transitorio, semidefinido, *estándar*.

Valores: producto cartesiano de los de los campos

record <*campos*> {;<*campos*> } **end**

<*campos*> ::= <*id-campo*> {,<*idcampo*> } : <*tipo*>

Operaciones:

- Acceso a cada componente por su nombre (`{$R+$}` ó `-Cr`)

  <*id-registro*> .<*id-campo*>
- Para cada componente las de su tipo
- *Asignación*
- Paso como parámetro *(tipo con nombre)*
- Las que el programador defina

Slide 37

```
type
  tfecha = record
                dia : 1..31;
                mes : 1..12;
                anyo : 1900 .. 2100;
           end;
  tcalif = (np, su, ap, no, ss, mh);
  talumno = record
            dni, num : integer;
            fnac, fmat : tfecha;
            nota : array [1..NNOTAS] of real;
            notam : real;
            calif : tcalif;
         end;
  tpersona = record
           nombre : array [1..LON] of char;
           sexo : (hombre, mujer);
           estado :(sol, cas, div, viu, cle);
           fechanac : tfecha;
         end;
   var alumnos : array [1.. NAL] of talumno;
       al : talumno;   uno, otro : tpersona;
```

Slide 38

```
readln (al.num)
al.nota[1] := 7;
al.notam := suma/NNOTAS;
al.fnac.dia := 12;
uno.sexo := hombre;
uno.fnac := al.fnac;
alumnos[3]:= alumno;
if alumno[i].nomtam > 5 then ...
```

Slide 39

```
function edad (x: tpersona, hoy: tfecha) integer;
begin
   if (hoy.mes > x.fechanac.mes) or
      (hoy.mes = x.fechanac.mes)
      and (hoy.dia >= x.fechanac.dia)
      then edad := hoy.anyo-x.fechanac.anyo
      else edad := hoy.anyo-x.fechanac.anyo-1
end;
```

Slide 40

```
procedure leerfecha (var f : TFecha);
begin
   writeln ('Escriba dia, mes, año: (2 2 2000)');
   readln (f.dia, f.mes, f.anyo)
end;

procedure escribirfecha (f : TFecha);
begin
   write ('Día ', f.dia:2, ' de ');
   case  f.mes of
       1 : write ('enero');
       ...
      12: write ('diciembre')
   end; (* case *)
   writeln (' de ', f.anyo:4)
end;
```

Slide 41

```
function indice
   (var v:talumnos; idesde: integer; n: integer):integer;
(* Devuelve el índice del alumno con número n *)
(* Pre: n está en la lista v
        a partir de la posición idesde *)
var i: integer;
begin
    i := idesde;
    while v[i].num <> n do i := i+1;
    indice := i
end;
```

Slide 42

```
function ibusca
   (var v:talumno, idesde, ihasta:integer;
    n:integer): integer;
(* Devuelve la posición del alumno de num n en v
   Si no está, devuelve ihasta+1
   Pre: 1<=idesde<=ihasta<=NAL *)
var i : integer
begin
   i := idesde;
   while (i<ihasta) and (v[i].num <> n) do i := i+1;
   if v[i].num = n then ibusca := i
                   else ibusca := ihasta +1;
end;
```