Registros

Programación 1

InCo - FING

Section 1

Tipos en Pascal

Clasificación de tipos

- **Elementales**
  - Ordinales o escalares: `Integer`, `Char`, `Boolean`, subrangos, enumerados.
  - Otros: `real`
- **Estructurados**:
  - arreglos (`array`)
  - conjuntos (`set`)
  - registros (`record`)

Tipos estructurados

- **Arreglos**: Secuencias *homogéneas* de elementos. El orden es relevante y puede haber repetidos. Cantidad fija de elementos.
- **Conjuntos**: Conjuntos *homogéneos*, sin orden, no hay repetidos. La cantidad de elementos es variable pero **acotada**.
- **Registros**: Colección *heterogénea* de datos. Cantidad **fija** de elementos.

Section 2

Tipo Registro

Motivación

Es usual la representación de *objetos* o *entidades* de la realidad mediante estructuras de datos.

Una de las estructuras más utilizadas con este fin es el *record*.

Un *record* es una tupla de datos de diferente tipo, cada uno de los cuales se accede mediante un nombre de *campo*.

Por ejemplo,

	CI	Nombre	Direc.	Tel.
estudiante	4235678	Juan Pomi	Yi 2345	6718990

Definición de un registro

```
type
        T = record
                campo1 : tipo1;
                campo2 : tipo2;
                ...
                campok : tipok
            end;
```

Donde *campo-i* es un identificador que da nombre a un campo y *tipo-i* es su correspondiente tipo.

Ejemplo(1)

Datos de un estudiante.

```
type
    TCedula   = array [1..8] of 0..9;
    TNombre   = array [1..45] of char;
    TTelefono = array [1..7] of 0..9;
    TDirec    = array [1..40] of char;

    TEstudiante = record
                      cedula    : TCedula;
                      nombre    : TNombre;
                      telefono  : TTelefono;
                      direccion : TDirec
                  end;
```

Ejemplo(2)

Los puntos del plano pueden representarse como una pareja de reales:

```
type
    punto = record
                coordenadaX,
                coordenadaY : real
            end;
```

Ejemplo(3)

Los números racionales pueden representarse así:

```
type
     TSigno   = (mas,menos);
     natural  = 0..MaxInt;
     positivo = 1..MaxInt;

     racional = record
          signo       : TSigno;
          numerador   : natural;
          denominador : positivo;
     end;
```

Acceso a campos de registros

El operador . (punto) seguido del nombre del campo se utiliza para acceder a los componentes de un registro.

```
var q : racional;

      ...
      q.signo:= mas;
      q.numerador:= 4;

      writeln(q.denominador);
      ...
```

Lectura de un registro

No es posible aplicarle `read` a una variable de un tipo registro.

```
procedure LeerRacional(var q : racional);
var
   signo : -1..+1;
begin
    write('Ingrese signo(-1,0,+1): ');
    readln(signo);
    if signo = -1 then
       q.signo:= menos
    else
       q.signo:= mas;
    write('Ingrese denominador: ');
    readln(q.denominador);
    write('Ingrese numerador: ');
    readln(q.numerador);
end; {LeerRacional}
```

Escritura de un registro

Para mostrar un registro se debe desplegar campo por campo.

```
procedure MostrarRacional(q :racional);
begin
    if q.signo = menos
       then write('-');        (* signo *)

    write(q.numerador);        (* numerador *)

    write('/');                (* separador *)

    write(q.denominador);      (* denominador *)
end; {MostrarRacional}
```

Ejemplo: suma de dos racionales.

```
function SignoRac(q: racional): integer;
begin
  case q.signo of
    mas   : SignoRac:= +1;
    menos : SignoRac:= -1;
  end
end; {SignoRac}

procedure SumaRacionales(p,q: racional; var resultado: racional);
var
   num: integer;
begin
    resultado.denominador:= q.denominador * p.denominador;
    num:= SignoRac(p) * p.numerador * q.denominador
          +
          SignoRac(q) * q.numerador * p.denominador;

    resultado.numerador:= abs(num);
    if num < 0 then
       resultado.signo:= menos
    else
       resultado.signo:= mas
end; {SumaRacionales}
```

El tipo Fecha

No existe un tipo **fecha** predefinido en Pascal.

```
type
    TFecha = record
        anio: 0..10000;
        mes : 0..12;
        dia : 1..31;
    end;
```

Comparación de Fechas

```
function AnteriorFecha(f1,f2: TFecha) : boolean;
begin
    if f1.anio = f2.anio then
       if f1.mes = f2.mes then
          AnteriorFecha:= f1.dia < f2.dia
       else
          AnteriorFecha:= f1.mes < f2.mes
    else
       AnteriorFecha:= f1.anio < f2.anio;
end; {AnteriorFecha}
```

La instrucción `with`

La instrucción `with` permite simplificar la forma de referenciar los campos de un registro.

```
procedure MostrarRacional(q :racional);
begin
    with q do
    begin
        if signo = menos
           then write('-');     (* signo *)
        write(numerador);       (* numerador *)
        write('/');             (* separador *)
        write(denominador);     (* denominador *)
    end; {with}
end; {MostrarRacional}
```

Resolución de ambigüedad con `with`

```
var
   q     : racional;
   signo: integer;

   ...
    with q do
    begin
      ...
      signo:= 1;   (* error!  representa q.signo *)
      ...
      signo:= mas; (* correcto *)
      ...
    end;
```

Asignación de registros

Una variable de tipo registro se puede asignar a otra del mismo tipo.

```
var
   p1,p2: punto;
    ...
    p1 := p2;   (* asignación campo a campo *)

    ...
```

Estructuras complejas

Anidamiento de tipos estructurados.

```
type
    TFechas = array [1..Max] of fecha;
    ...
begin
    ...
    (* campo anio de la séptima celda *)
    a[7].anio := 1968;
    ...
```