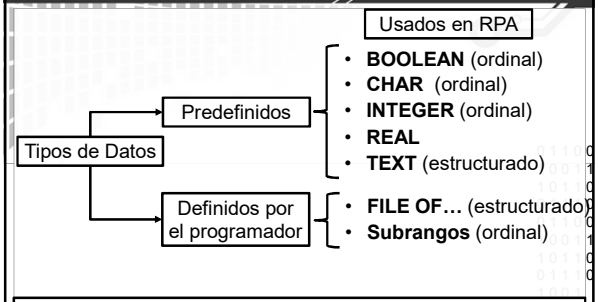
RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

CLASE 13

Definición y compatibilidad de tipos de datos.
Sentencia condicional CASE.

Luciano H. Tamargo
http://cs.uns.edu.ar/~lt
Depto. de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca
2016

CONCEPTOS: TIPOS DE DATOS EN PASCAL



- Definir nuevos tipos de datos permite claridad y abstracción.
- Dos conceptos fundamentales en el desarrollo de Software.

TIPOS DEFINIDOS POR EL PROGRAMADOR

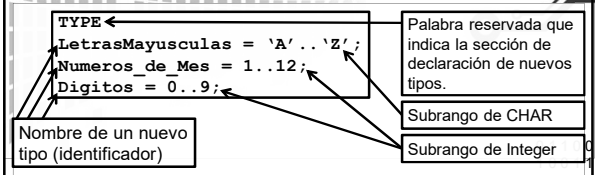
- Poder definir y usar tipos de datos fue un muy importante avance en la evolución de los lenguajes de programación.
- Permiten dar **claridad** al código fuente. Esto ayuda al programador a leer el código y entonces **prevenir errores** de programación.
- También dan **información** al **compilador**, que puede ser usada para **prevenir errores**, y además generar un **mejor código** ejecutable.
- Hay compiladores que realizan un **chequeo de tipos** al compilar, otros al ejecutar, y otros en ambos momentos (en algunos casos se puede configurar mediante opciones al compilar).

TIPOS DEFINIDOS POR EL PROGRAMADOR

IMPORTANTE:

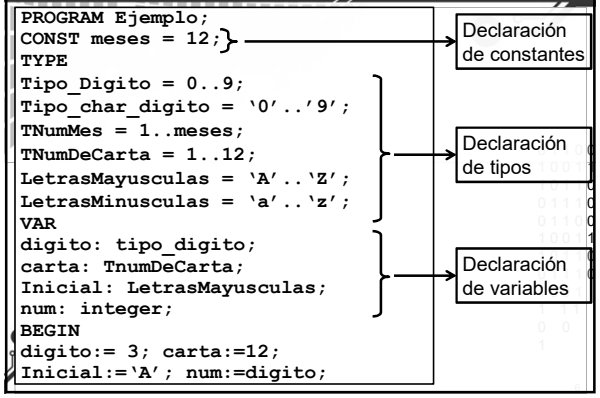
- En esta materia vamos a usar solamente algunas de las ventajas de poder definir nuevos tipos.
- Más adelante, en otras materias de su carrera descubrirá muchas más ventajas del uso de tipos definidos por el programador.

TIPOS DEFINIDOS POR EL PROGRAMADOR: SUBRANGOS



- Se puede definir **tipos subrangos**, de cualquier otro **tipo ordinal**.
- Se indica: un **identificador** como nombre del nuevo tipo, luego el símbolo "**=**", y finalmente, separados por un par de puntos consecutivos "**..**" un **valor inicial** y un **valor final** de algún tipo ordinal.
- Estos valores definen el "RANGO" de todos los valores posibles para los elementos de este nuevo tipo de dato.
- Las operaciones son las mismas que las del tipo ordinal del cual se hace el subrango.

NUEVOS TIPOS DE DATOS DEFINIDOS POR EL PROGRAMADOR



OTRA FORMA DE DEFINIR NUEVOS TIPOS DE DATOS

```
TYPE
Archivo_Letras = FILE OF char;
Archi_temperaturas = FILE OF real;
componentes = integer;
Productos = FILE OF componentes;
```

Esto será necesario para conceptos de se verán en las próximas clases.

```
VAR
Letras: Archivo_letras;
Elem: componente;
Stock: Productos;
```

- En Pascal también se pueden definir **nuevos tipos (o nuevos nombres para un tipo)** en función de tipos ya existentes.
- Se indica: un identificador como nombre para el nuevo tipo, luego el símbolo "=", y luego algún tipo predefinido, tipo estructurado, o tipo definido por el usuario antes.
- Los valores posibles y las operaciones disponibles de este nuevo tipo de dato, son las mismas del tipo usado a la derecha del "=".

TIPOS DEFINIDOS POR EL PROGRAMADOR

```
PROGRAM Ejemplo;
TYPE
 Entero = integer;
 Logico = Boolean;
 NroReal = real; Letra = char;
 Telemento = Entero;
 TipoArchivo = FILE OF Telemento;
VAR
 Inicial: Letra; Es_Par: Logico;
 Num: Telemento ;
 Archivo1, Archivo2: TipoArchivo;
BEGIN
 Inicial := 'A'; Num:= 4;
 Es_Par := (Num MOD 2 ) = 0
 Assign(Archivo1,'num.dat');
 rewrite(archivo1);
 write(Archivo1,num);
```

Declaración de tipos

Declaración de variables

RELACIONES ENTRE TIPOS DE DATOS

- **En Pascal** existen tres **relaciones entre tipos de datos**:

1- Igualdad o Identidad

2- Compatibilidad

3- Compatibilidad de asignación

1- TIPOS IDÉNTICOS EN PASCAL

- **Dos elementos** tienen **tipos idénticos** si se cumple una de las siguientes opciones:
  a) Están declarados con el mismo identificador de tipo.
  b) Los identificadores de tipo son diferentes (ej: T1 y T2) pero han sido definidos como equivalentes por una declaración de la forma T1 = T2.
- **Ejemplo**: ¿Cuáles variables tienen tipos idénticos?

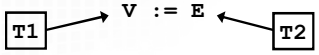
```
TYPE
 T = INTEGER;
 T1 = T;
VAR
 A, A1: T;
 A2: REAL;
 B: INTEGER;
 C: T1;
 D: -32768 .. 32767;
```


2- TIPOS COMPATIBLES EN PASCAL

- Dos tipos son **compatibles** si al menos una de las siguientes opciones es verdadera:
  a) Ellos son idénticos.
  b) Uno es subrango del otro.
  c) Ambos son subrangos del mismo tipo.
- **Ejemplo**: ¿Cuáles son compatibles?

```
TYPE
 T = Integer;
 Sub = 1..1000;
 Sub2 = Sub;
 Sub1 = 100..200;
 Sub3 = 0..99;
 TipoNum = Real;
VAR
 A: Sub;
 B: INTEGER;
 C: Sub1;
 D: Sub2;
 E: Sub3;
 F: TipoNum;
```


3- COMPATIBILIDAD DE ASIGNACIÓN



- Una expresión **E** de tipo **T2** es **asignación-compatible** con el identificador **V** de tipo **T1** si al menos una de las siguientes declaraciones es verdadera:
  1. **T1** y **T2** son idénticos.
  2. **T1** es real y **T2** es entero o subrango de entero.
  3. **T1** y **T2** son subrangos o enteros, y el valor de **E** es un valor permitido del tipo **T1**.

SENTENCIA CONDICIONAL: CASE

SENTENCIAS CONDICIONALES EN PASCAL

Problema propuesto: Escriba un programa en Pascal que lea un valor char, y si es "@" indique en pantalla "arroba"; si es un dígito '0' a '9' indique "dígito"; si es un operador de suma, resta, multiplicación o división, indique "operador"; si es una letra mayúscula o minúscula, indique "letra".

Parte de una posible solución

```
IF valor = '@' THEN
  write(' arroba ')
ELSE
  IF (valor >= '0') and (valor <= '9') THEN
    writeln(' dígito')
  ELSE
    IF (valor = '+') or (valor = '-') or
       (valor = '*') or (valor = '/') THEN
      writeln(' operador ')
    ELSE
      IF (valor >='A') and (valor <='Z') or
         (valor >= 'a') and (valor<= 'z') THEN
        writeln(' letra');
```

SENTENCIA CONDICIONAL CASE EN PASCAL

- **CASE** es un sentencia condicional que permite discriminar para distintos "casos" (valores) que sentencia debe ejecutarse.

Aquí se especifica un solo caso.

Aquí se especifican 10 casos

Aquí se especifican 4 casos.

Aquí se especifican 52 casos (2x26)

```
PROGRAM opciones; //reconoce símbolos
VAR valor: char;
BEGIN
 write('ingrese valor ASCII');
 readln(valor);
 CASE valor OF
  '@': write('arroba');
  '0'..'9': writeln('dígito');
  '+','-','*','/':writeln('operador');
  'A'..'Z','a'..'z':writeln('letra');
 END;
 write('fin del programa');
 readln;
END.
```


SENTENCIAS CONDICIONALES EN PASCAL

- Una sentencia **CASE** puede considerarse como una "abreviatura" de varios **IF-THEN-ELSE** anidados.
- Todo **CASE** puede reescribirse con **IF-THEN-ELSE** anidados.
- Por ejemplo, el case anterior puede reescribirse.

```
IF valor = '@' THEN
  write(' arroba ')
ELSE
  IF (valor >= '0') and (valor <= '9') THEN
    writeln(' dígito')
  ELSE
    IF (valor = '+') or (valor = '-') or
       (valor = '*') or (valor = '/') THEN
      writeln(' operador ')
    ELSE
      IF (valor >='A') and (valor <='Z') or
         (valor >= 'a') and (valor<= 'z') THEN
        writeln(' letra');
```

SENTENCIA CASE (SINTAXIS)

```
CASE <expresion> OF
<lista_opciones>: <una sentencia simple o compuesta>;
<lista_opciones>: <una sentencia simple o compuesta>;
    ...
END; {sugerencia: vea el diagrama sintáctico de CASE}
```

1. <expresion> cualquier expresión que **sea de tipo ordinal**
2. <lista_opciones> puede ser:
   a) un valor individual. Ej: 2
   b) valores individuales separados por coma. Ej: 2,5,7
   c) Rangos de valores. Ej: 1..100
   d) una combinación de (b) y (c) ej: 1..10, 13, 15..20
3. Las listas de opciones **deben ser disjuntas**. No puede haber opciones repetidas, **es un error de compilación**

SENTENCIA CASE (SEMÁNTICA)

```
CASE <expresion> OF
<lista_opciones>: <una sentencia simple o compuesta>;
<lista_opciones>: <una sentencia simple o compuesta>;
    ...
END;
```

- <expresion> y <lista_opciones> deben ser del mismo tipo
  1. Se evalúa <expresion> y se obtiene un **valor**.
  2. Se busca (de arriba hacia abajo) **valor** está en una de las <lista_opciones>.
  3. Si se encuentra el **valor** se ejecuta la sentencia siguiente al ":" y luego sigue en el **END**;
  4. Si **valor no pertenece** a ninguna de las <lista_opciones> no se ejecuta ninguna sentencia.

OPCIONES DE UNA SENTENCIA CASE

Una expresión (ordinal)

Puede haber un único valor en la opción

Puede haber varios separados por comas

Pueden haber un rango de valores

Pueden haber una combinación de valores y rangos

```
CASE trunc(R)-3*2 OF
  4 : BEGIN
        …sentencias
        …
      END;
  1,2,3: write(' 1 2 o 3');
  50..100: WRITE(' 5 a 10 ');
  101,201, 300..400,
  501..1001, 2001: BEGIN
                     …
                   END;
END; {del case}
```


OBSERVACIONES SOBRE SENTENCIA CASE

- **Opciones repetidas**

```
VAR
  M: INTEGER;
CASE M OF
  1, 5 :  <sentencia>
  5  3 :  <sentencia>
  4..10:  <sentencia>
END;
```

MAL

- No puede haber opciones repetidas, es un error de compilación.
- Las listas de opciones deben ser **disjuntas.**

OBSERVACIONES SOBRE SENTENCIA CASE

- **Extensión a Pascal estándar**

```
VAR
  M: INTEGER;
CASE M OF
  -9..9: write(' 1 dígito');
  -99..-11,11..99: write(' 2 dígitos');
ELSE
  write(' más de 2 dígitos');
END;
```

- el **ELSE** se ejecuta cuando el valor no corresponde a ninguna opción

OTRA SOLUCIÓN PARA "DÍAS DE UN MES" (USANDO CASE)

```
VAR
 mes, anio, cant_dias: INTEGER;
CASE MES OF
  11,4,6,9: cant_dias := 30;
  2: IF (anio mod 4=0) and (anio mod 100<>0) or
        (anio mod 400=0) THEN
      cant_dias := 29
    ELSE
      cant_dias := 28;
  1,3,5,7,8,10,12: cant_dias :=31;
END; {--- fin del case --- }
Writeln('Tiene', cant_dias,' días');
END.
```


FUNCIONAMIENTO DE CASE EN PASCAL

```
…
readln(mes, anio);
CASE MES OF
  11,4,6,9: cant_dias := 30;
  2: IF (anio mod 4=0) and
        (anio mod 100<>0) or
        (anio mod 400=0) THEN
      cant_dias := 29
    ELSE
      cant_dias := 28;
  1,3,5,7,8,10,12: cant_dias :=31;
END; {--- fin del case --- }
Writeln(cant_dias);
END.
```

1. si el valor de MES está entre estos valores
2. entonces se ejecuta esta sentencia,
3. una vez que se ejecuta una opción se pasa a la sentencia que sigue al case

FUNCIONAMIENTO DE CASE EN PASCAL

```
…
readln(mes, anio);
CASE MES OF
  11,4,6,9: cant_dias := 30;
  2: IF (anio mod 4=0) and
        (anio mod 100<>0) or
        (anio mod 400=0) THEN
      cant_dias := 29
    ELSE
      cant_dias := 28;
  1,3,5,7,8,10,12: cant_dias :=31;
END; {--- fin del case --- }
Writeln(cant_dias);
END.
```

si el valor de MES NO está entre estos valores

pasa a la siguiente opción, y así sucesivamente...

Si MES tiene un valor que no figura en ninguna de las opciones, entonces no se ejecuta ninguna opción del case.

PROBLEMA PROPUESTO

- Un día es:
  - **muy frío** si la temperatura máxima está entre -20 y 1 grado,
  - **frío** si su máxima está entre 2 y 10,
  - **templado** si está entre 11 y 20,
  - **cálido** entre 21 y 28 y
  - **muy caluroso** entre 29 y 45.
- Considere un archivo de enteros 'temperaturas.dat' que tiene las temperaturas máximas de un mes.
- Escriba un programa que calcule cuantos días muy fríos, fríos, templados, cálidos, y muy calurosos ocurrieron en ese mes.

SENTENCIA CASE. EJEMPLO

```
CASE  …  OF
   -20..1:  … muy frío ….
   2..10:   … frío …
   11..20:  … Templado ….
   21..28:  … cálido …
   29..45:  … muy caluroso …
END
```