Resolución de Problemas y Algoritmos

Clase 13:
Definición y compatibilidad de tipos de datos.
Sentencia condicional CASE.

Charles Babbage

Dr. Alejandro J. García
http://cs.uns.edu.ar/~ajg

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

Observaciones importantes

- **Los temas sobre los cuales trabajaremos a partir de hoy:**
  - **serán evaluados en el segundo parcial,**
  - **y dado que no tendrán la práctica suficiente, no se espera que los usen para resolver problemas del primer parcial.**
- **Todos los temas que ya hemos trabajado y que se evalúan en el primer parcial se seguirán usando como base para los temas que veremos a partir de hoy.**

Conceptos: Tipos de datos en Pascal

Ejemplos de tipos que se usan en RPA:

Se pueden dividir en:

Tipos de datos

Predefinidos
- **BOOLEAN** (ordinal)
- **CHAR** (ordinal)
- **INTEGER** (ordinal)
- **REAL**
- **TEXT** (estructurado)

Definidos por el programador
- **FILE OF…** (estructurado)
- **Subrangos** (ordinal)

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 al 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 mediantes opciones al compilar).

IMPORTANTE:

- En esta materia vamos a usar solamente algunas de las ventajas 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

```
TYPE
  LetrasMayusculas = 'A' .. 'Z';
  Numeros_de_Mes = 1..12;
  Digitos = 0..9;
```

Palabra reservada que indica la sección de declaración de nuevos tipos.

Subrango de CHAR

Subrangos de INTEGER

Nombre de un nuevo tipo (identificador)

- 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 haceel subrango.

Nuevos tipos de datos definidos por el programador

```
PROGRAM Ejemplo;
CONST meses=12;
TYPE
  Tipo_Digito = 0..9;
  Tipo_char_digito = '0'..'9';
  TNumMes = 1..meses;
  TNumDeCarta = 1..12;
  LetrasMayusculas = 'A' .. 'Z';
  LetrasMinusculas = 'a' .. 'z';
VAR
  digito: tipo_digito;  carta: TnumDeCarta;
  Inicial: LetrasMayusculas; num: integer;
BEGIN
  digito:= 3; carta:=12; Inicial:='A'; num:=digito;
```

Declaración de constantes

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.

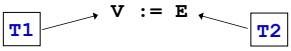
b) Uno es subrango del otro.

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".

```
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');
```

Parte de una posible solución

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 pertence** 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

Pueden 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)

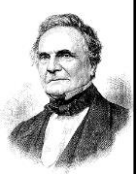
```
mes, anio, cant_dias: INTEGER;

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.
```


Charles Babbage (1791-1871)

Matemático y científico de la computación británico.

En 1812 intentó encontrar un método por el cual se pudieran **hacer cálculos automáticamente por una máquina a vapor**.



Eliminando así los errores debidos a la fatiga o aburrimiento que sufrían las personas encargadas de compilar las tablas matemáticas de la época.

Tres diversos factores parecían haberlo influido: su aberración al desorden, su conocimiento de tablas logarítmicas, y los trabajos de máquinas calculadoras realizadas por Blaise Pascal y Gottfried Leibniz.

http://es.wikipedia.org/wiki/Charles_Babbage

La máquina de las diferencias de Babbage

Diseñó y parcialmente implementó una máquina a vapor de diferencias mecánicas para calcular tablas de números (ver).

Parte de su máquina original.

Babbage's difference engine, construida por el Museo de Ciencias Británico en 1991 usando el plano original.

Hay varios videos en YouTube que muestran la máquina funcionando

La máquina analítica de Babbage

Entre 1833 y 1842, Babbage diseño e intentó construir una máquina que fuese programable para hacer cualquier tipo de cálculo. El diseño se basaba en el telar de Joseph Jacquard, el cual usaba tarjetas perforadas para determinar como una costura debía ser realizada.

Se considera que la máquina analítica de Babbage fue la primera computadora. Un diseño inicial plenamente funcional de ella fue terminado en 1835. Pero la máquina nunca se construyó (hasta ahora…)

El Museo Británico tiene un proyecto para construirla y esperan terminar para 2021 (150 años después de la muerte de Babbage). Sería equivalente a una computadora con un reloj de 7 Hz y con 675 bytes de memoria.

http://www.sciencemuseum.org.uk/onlinestuff/stories/babbage.aspx

Volver