

Curso de Programación 1

Plan 97

Clase 7

(Funciones, Recursión, Tipos Enumerados)

Funciones en Pascal

Funciones son subprogramas que computan un valor único. Por ejemplo, la función estándar `sqrt` computa la raíz cuadrada de un número no negativo.

A continuación discutiremos los dos tipos de funciones que Pascal maneja,

- las primitivas y
- las definidas por el usuario.

Funciones primitivas (estándar)

Son las funciones disponibles en Pascal estándar.

Ya hemos visto `sqrt`, `sqr`, `abs`, `round`, `trunc`.
Otras son: `sin`, `cos`, `arctan`, `exp`, `ln`, `odd`.

También están incluidas `pred`, `succ`, `ord` y `chr`
que discutiremos más adelante.

Funciones definidas por el usuario

Las **funciones** son similares a los procedimientos excepto que el objetivo de una función es devolver únicamente un valor al punto en que se llamó.

```
function <nombre> (<parámetros>): <tipo>
<declaraciones locales>
begin
    <sentencias>
end
```

El valor que se devuelve es **asignado** al nombre de la función.

Ejemplo

Declaración

```
function par (n: integer) : boolean;  
begin  
    par := n mod 2 = 0;  
end;
```

Invocación

```
readln (x);  
if par(x) then ... else ...
```

Uso de funciones

El cuerpo de toda función debe contener al menos una asignación de valor al nombre de la función. Se pueden hacer varias asignaciones pero sólo se devolverá el último valor asignado.

El nombre de una función **no** actúa como una variable.

Usar el nombre de una función en una expresión no representa el último valor asignado al nombre, sino que denota una invocación a la función.

Ejemplo

La siguiente declaración de función no es correcta:

```
function pot10(x : integer) : integer;
var i : integer;
begin
    pot10 := 1;
    for i := 1 to 10 do
        pot10 := pot10 * x
    end
end
```

Ejemplo

La forma correcta de declararla es la siguiente:

```
function pot10(x : integer) : integer;
var i,temp : integer;
begin
    temp := 1;
    for i := 1 to 10 do
        temp := temp * x;
    pot10 := temp
end
```

Criterios de uso

Dado que una función retorna un único valor, como forma de enfatizar la relación entre funciones matemáticas y las definidas en Pascal, los parámetros de una función Pascal serán **siempre** pasados por valor.

En Pascal el tipo del resultado de una función sólo puede ser **simple** (no estructurado).

En caso que se requiera devolver más de un valor entonces debe usarse un procedimiento en lugar de una función.

Recursión

Una función o procedimiento puede invocarse a sí mismo. En dicho caso se está ante un procedimiento o función **recursivo**.

Ejemplo: La función **factorial**.

Definición matemática:

$$0! = 1$$

$$n! = n \times (n-1)!$$

Recursión

Implementación: Precondición $n \geq 0$

```
function factorial (n : integer) : integer;
begin
  if n <= 1
    then factorial := 1
    else factorial := n * factorial(n-1)
end;
```

Recursión

Ejemplo: Deseamos calcular la potencia n -ésima de un real x . Precondición $n \geq 0$.

```
function pot(x: real; n: integer): integer;
begin
  if n = 0
    then pot := 1
    else pot := x * pot(x,n-1)
end;
```

Recursión

Ejemplo: Deseamos eliminar los factores de 2 de un número natural positivo `num`. Vamos a calcular `val` y `n` tales que $\text{num} = 2^n \times \text{val}$, donde `val` es impar.

```
procedure elim(num:integer;
               var n,val:integer);
begin
  if par(num)
  then begin elim(num div 2,n,val);
         n := n + 1
       end
  else begin val := num; n := 0 end
end;
```

13

Recursión e Iteración

La función factorial también puede ser escrita sin hacer uso de recursión

```
function factorial(n : integer) : integer;
var i, fact : integer;
begin
  for i:= 1 to n
  do
    fact := fact * i;
  factorial := fact
end;
```

Programación 1. Plan 97. InCo. Fac. de Ingeniería

14

Recursión e Iteración

Si un lenguaje de programación como Pascal provee tanto recursión como iteración, cuál de los dos métodos debería ser usado para resolver un problema dado?

En una solución recursiva el computador debe controlar y resolver cada una de las invocaciones manteniendo información acerca del resultado las mismas.

Esto puede ser costoso en tiempo y espacio.

Recursión e Iteración

En general, se recomienda el uso de una solución recursiva sólo en caso que:

- la solución no puede ser fácilmente expresable iterativamente (sucede frecuentemente), o
- la eficiencia de la solución recursiva es satisfactoria.

Tipos Enumerados

Es frecuente encontrar casos en que un dato representa un valor entre un número finito, relativamente reducido, de alternativas.

Situaciones como esta se resuelven declarando un **tipo de dato enumerado** en el cual se definen todos los posibles valores alternativos.

```
Type T = (c1, ..., cn);
```

El usuario es quien le asigna un nombre a cada una de las constantes c_1, \dots, c_n .

Tipos Enumerados

Ejemplos:

Type

```
ColorPrim = (rojo, amarillo, azul);
```

```
Dia = (lunes, martes, miercoles, jueves,  
viernes, sabado, domingo);
```

```
Mes = (ene, feb, mar, abr, ..., nov, dic);
```

```
Palo = (trebol, diamante, corazon, pica);
```

Tipos Enumerados

Es posible incluir la definición de un tipo enumerado mismo en la declaración de una variable de ese tipo.

Ejemplo:

```
Var c: (rojo, amarillo, azul);
```

Valores de tipo enumerado pueden asignarse a variables del mismo tipo.

Ejemplo: `c := rojo;`

Tipos Enumerados

Las constantes c_i que aparecen en la definición de un tipo enumerado satisfacen una relación de orden ($<$) dada por la secuencia de enumeración.

Esto es,

$$c_i < c_j \Leftrightarrow i < j$$

Por ejemplo, para

```
Type Vocal = (A, E, I, O, U);
```

el orden es:

```
A < E < I < O < U
```

Tipos Enumerados

El orden < puede ser usado para formar expresiones booleanas que involucren valores del tipo enumerado.

Ejemplo:

```
Var feriado: Dia;
```

```
if feriado < sabado
  then
    writeln('El feriado cae entre semana')
  else
    writeln('El feriado cae fin de semana');
```

Tipos Enumerados

Valores de tipo enumerado también pueden ser comparados por igualdad.

Por ejemplo, para

```
v:Vocal
```

la siguiente es una sentencia válida:

```
if v = E
  then writeln('La vocal es E')
  else writeln('La vocal no es E');
```

Tipos Enumerados

Las constantes de un tipo enumerado de **ninguna manera** deben ser vistas o tratadas como caracteres o strings. O sea, por ejemplo,

A y 'A'
lunes y 'lunes'

son cosas **totalmente distintas**.

Importante: No es posible leer o imprimir valores de un tipo enumerado.