Programación II - 2011

Árboles(continuación)

Enunciado de Ejemplo

Se lee la información de las propiedades que posee una inmobiliaria. De cada propiedad se lee: código, precio, metros cuadrados y dirección.

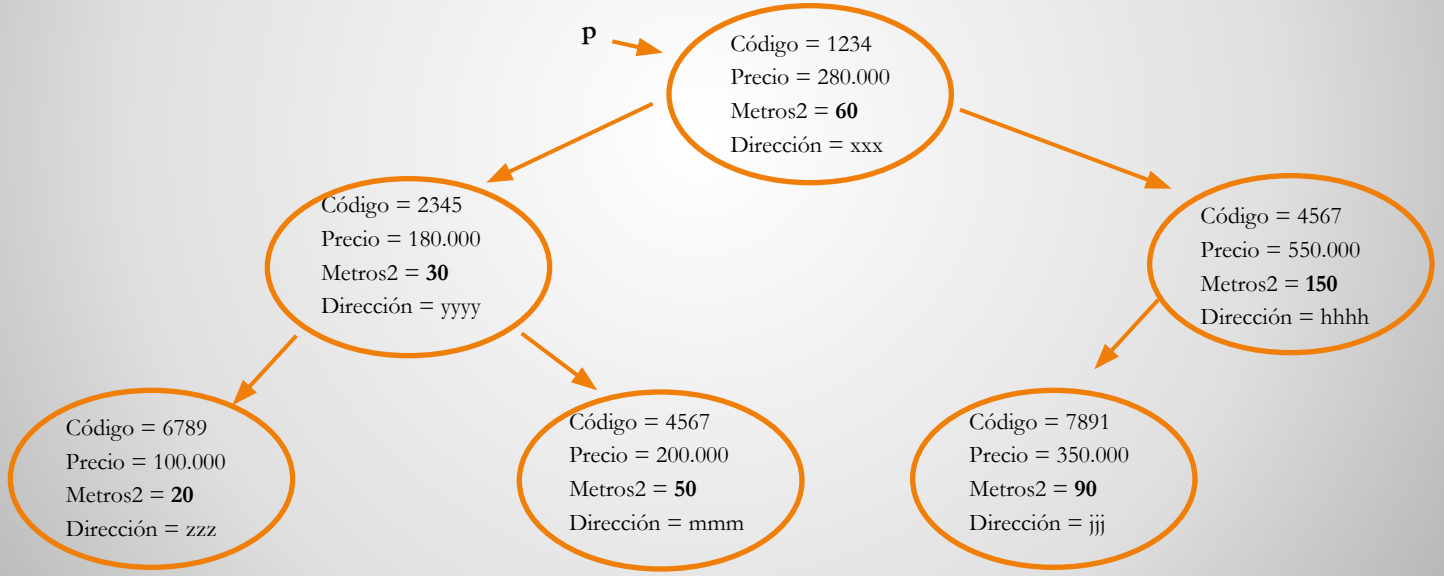
Se debe almacenar esta información en un árbol binario de búsqueda ordenado por metros cuadrados.

La lectura finaliza cuando se lee un código de propiedad igual a 0 (cero).
Una vez cargada la información, se desea calcular e informar:

1. El código de la propiedad con mayor cantidad de metros cuadrados.
2. La dirección de aquellas propiedades con menos de 72 metros cuadrados.
3. La dirección de aquellas propiedades con más de 25 y con menos de 55 metros cuadrados.
4. Si existe una propiedad con exactamente 72 metros cuadrados.
5. Código de la propiedad más costosa.

Ejemplo del ejercicio

Código = 1234 Precio = 280.000 Metros2 = 60 Dirección = xxx	Código = 2345 Precio = 180.000 Metros2 = 30 Dirección = yyyy	Código = 4567 Precio = 550.000 Metros2 = 150 Dirección = hhhh	Código = 4567 Precio = 200.000 Metros2 = 50 Dirección = mmm	Código = 6789 Precio = 100.000 Metros2 = 20 Dirección = zzz	Código = 7891 Precio = 350.000 Metros2 = 90 Dirección = jjj	Código = 0 Precio = Metros2 = Dirección =



Solución Propuesta

```
Program ejercicio;
CONST
        limite_inferior = 25;
        limite_superior = 75;
        metros = 72;
TYPE
        cadena = string[50];
        propiedad = record
           cod   : integer;
           precio: real;
           mts   : real;
           dir   : cadena;
         end;
        propiedades = ^nodo_arbol;

        nodoArbol  = record

           hi  : propiedades;
           dato: propiedad;
           hd  : propiedades;
        end;
```

Solución Propuesta

{Proceso que lee la información de las propiedades y va generando el árbol correspondiente.}

```
procedure generar_arbol(var p: propiedades);
var
   prop: propiedad;
begin
  leer_propiedad(prop);
  while (prop.cod <> 0) do begin
      agregar_en_arbol(p, prop);
      leer_propiedad(prop);
  end;
end;
```

Solución Propuesta

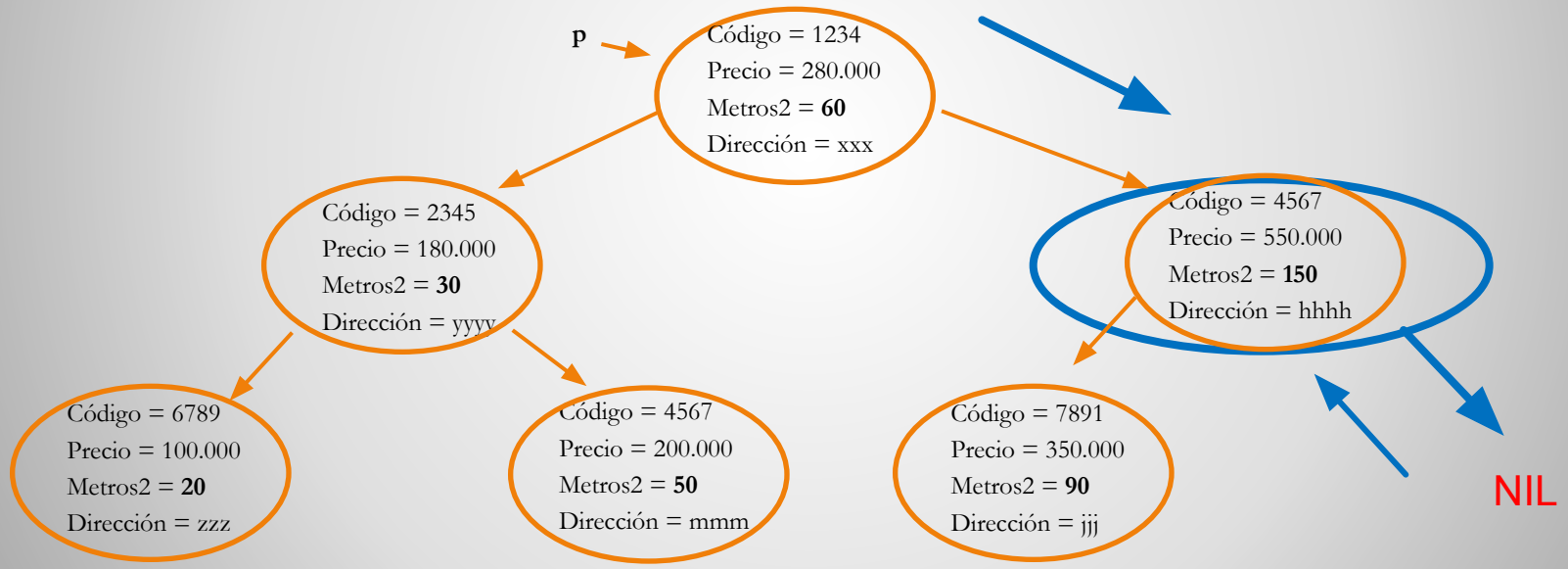
```
{Proceso que realiza la lectura de una propiedad}
procedure leer_propiedad(var p: propiedad);
begin
   writeln;
   writeln('----------Nueva Propiedad----------');
   writeln;
   write('Codigo: ');
   readln(p.cod);
   if (p.cod <> 0) then begin
      write('Precio: ');
      readln(p.precio);
      write('Metros 2: ');
      readln(p.mts);
      write('Direccion: ');
      readln(p.dir);
   end;
end;
```

Solución Propuesta

```
{Proceso que agrega una propiedad leída al árbol de propiedades}
procedure agregar_en_arbol(var p: propiedades; prop: propiedad);
begin
   if (p = nil) then begin
      new(p);
      p^.dato := prop;
      p^.hi   := nil;
      p^.hd   := nil;
   end
   else
      if (p^.dato.mts > prop.mts) then
            agregar_en_arbol(p^.hi, prop)
      else
       agregar_en_arbol(p^.hd, prop);
end;
```

Solución Propuesta

1. **El código de la propiedad con mayor cantidad de metros cuadrados.**



El código de la propiedad con mayor cantidad de metros cuadrados es 4567.

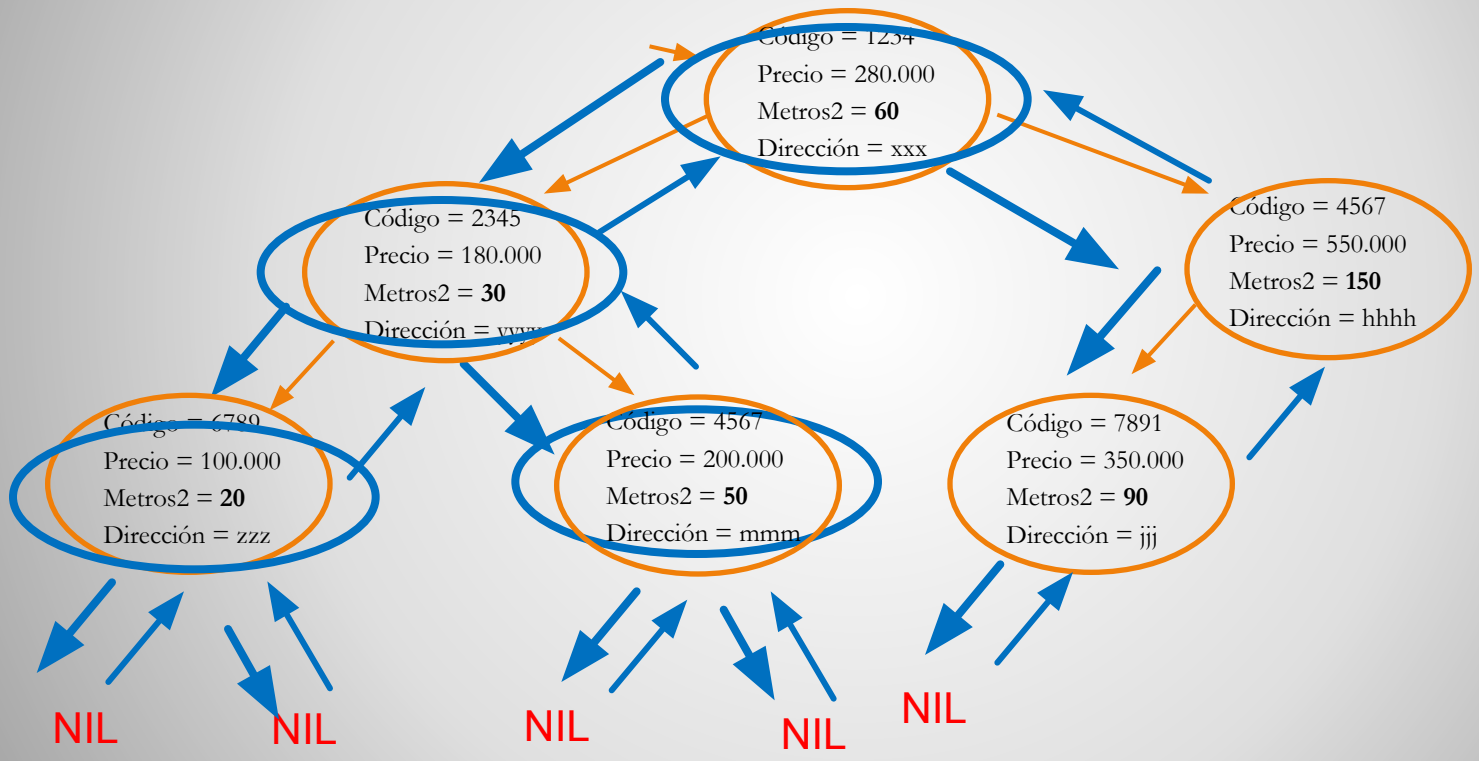
Solución Propuesta

```
{Función que retorna el código de propiedad con mayor metros cuadrados  }
{Suponemos que si es vacío retornamos el valor entero negativo –1}

function propiedad_mayor_mts(p: propiedades):integer;
begin
  if (p <> nil) then
     if (p^.hd = nil) then
         propiedad_mayor_mts := p^.dato.cod
     else
         propiedad_mayor_mts := propiedad_mayor_mts(p^.hd)
  else
        propiedad_mayor_mts := -1;
end;
```

Solución Propuesta

2. **La dirección de aquellas propiedades con menos de 72 metros cuadrados.**



Direcciones de propiedades con menos de 72 m2: ‘xxxx’, ‘yyy’, ‘zzz’, ‘mmm’

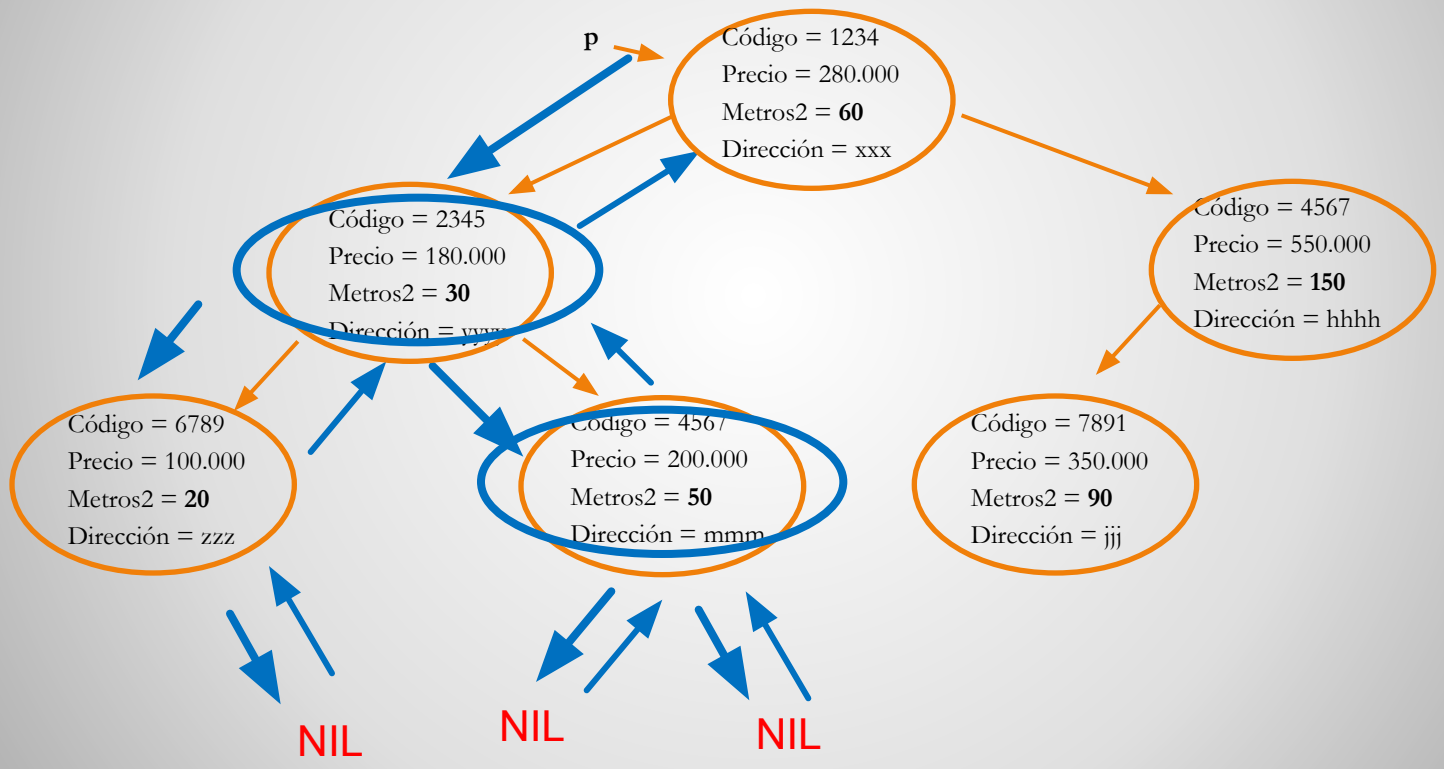
Solución Propuesta

{Proceso que informa la dirección de aquellas propiedades con menos de una cantidad de metros cuadrados recibida como parámetro)

```
procedure informar_direcciones_b(p: propiedades; metros: real);
begin
  if (p <> nil) then
    if (p^.dato.mts < metros) then begin
      writeln(p^.dato.dir);
      informar_direcciones_b(p^.hi, metros);
      informar_direcciones_b(p^.hd, metros);
    end
    else
      informar_direcciones_b(p^.hi, metros);
end;
```

Solución Propuesta

3. ***La dirección de aquellas propiedades con más de 25 y con menos de 55 metros***



Direcciones de prop. con mas de 25 y menos de 55 m2: ‘yyy’, ‘mmm’

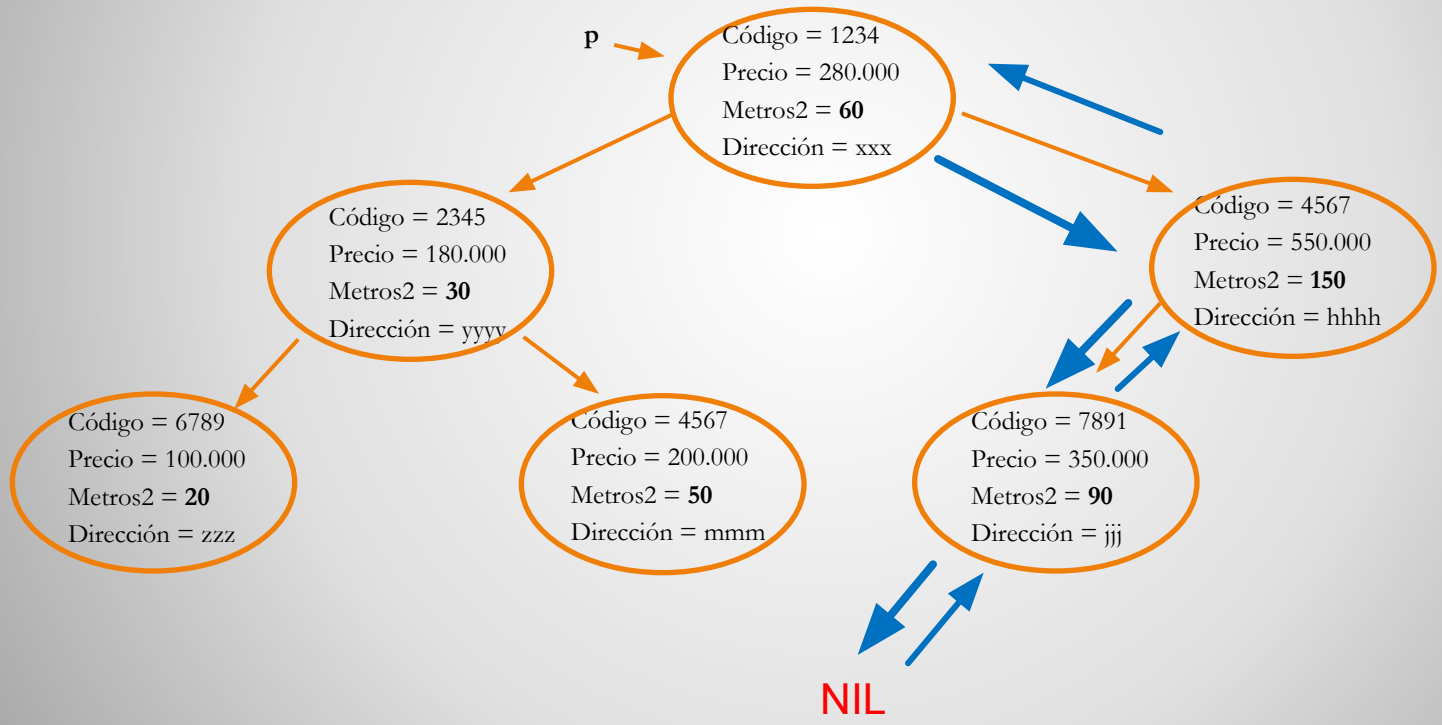
Solución Propuesta

{Proceso que informa la dirección de aquellas propiedades con más de 25 y con menos de 55 metros cuadrados }

```
procedure busqueda_acotada(p: propiedades; inf: real; sup: real);
begin
  if (p <> nil) then
    if (p^.dato.mts >= inf) then
      if (p^.dato.mts <= sup) then begin
        writeln(p^.dato.dir);
        busqueda_acotada(p^.hi, inf, sup);
        busqueda_acotada(p^.hd, inf, sup);
      end
      else
        busqueda_acotada(p^.hi, inf, sup)
    else
      busqueda_acotada(p^.hd, inf, sup);
end;
```

Solución Propuesta

4. ***Si existe una propiedad con exactamente 72 metros cuadrados.***



No existe ninguna propiedad con 72 metros cuadrados

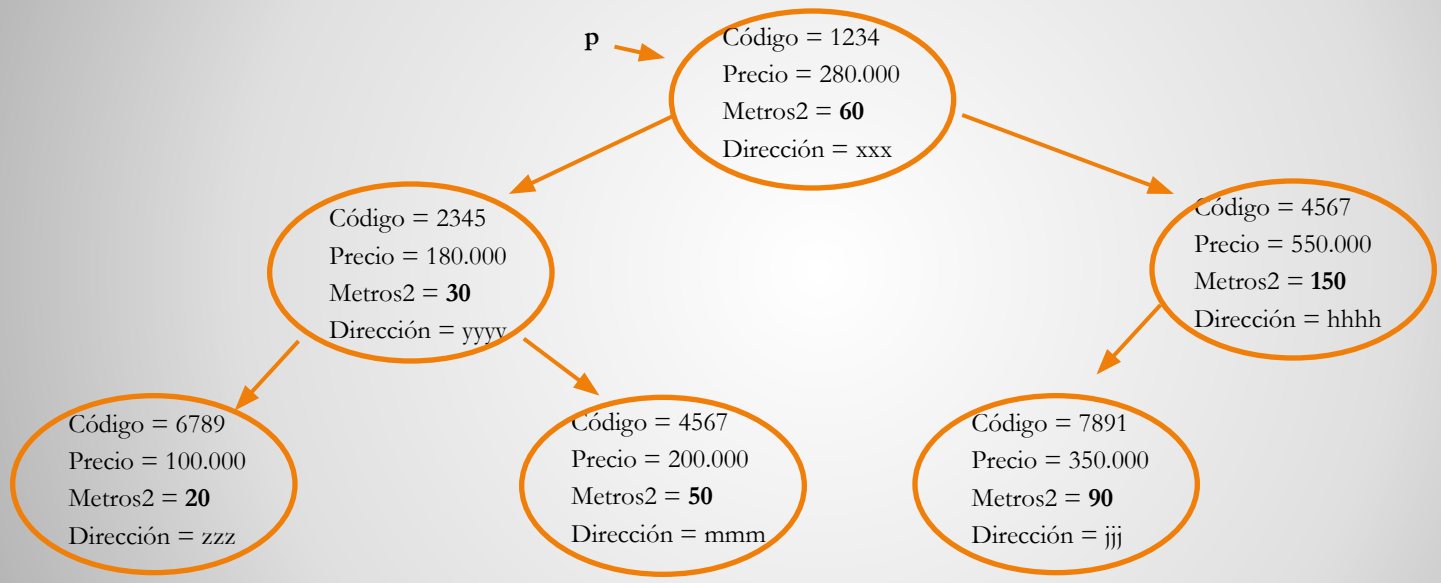
Solución Propuesta

{Función que retorna si encuentra alguna propiedad que posea exactamente una cantidad de metros cuadrados recibida como parámetro }

```
function existe_propiedad(p: propiedades; metros: real): boolean;
begin
  if (p <> nil) then
      if (p^.dato.mts > metros) then
          existe_propiedad := existe_propiedad(p^.hi, metros)
      else
          if (p^.dato.mts < metros) then
              existe_propiedad := existe_propiedad(p^.hd, metros)
          else
              existe_propiedad := true
  else
      existe_propiedad := false;
end;
```

Solución Propuesta

5. ***Código de la propiedad más costosa***



Cuando debemos recorrer el árbol por un atributo diferente al de su orden, debemos pasar por todos los nodos

Solución Propuesta

```
{Función que retorna el código de la propiedad más cara }
function propiedad_mas_costosa(p: propiedades): integer;
var
  cod_max : integer;
  precio_max: real;
begin
  precio_max := -1;
  cod_max    := -1;
  obtener_mayor_costo(p, cod_max, precio_max);
  propiedad_mas_costosa := cod_max;
end;
```

Solución Propuesta

```
{Proceso que obtiene la propiedad que contiene el precio máximo}
procedure obtener_mayor_costo(p: propiedades; var cod_max: integer; var precio_max: real);
begin
  if (p <> nil) then begin
    if (p^.dato.precio > precio_max) then begin
      precio_max := p^.dato.precio;
      cod_max    := p^.dato.cod;
    end;
    obtener_mayor_costo(p^.hi, cod_max, precio_max);
    obtener_mayor_costo(p^.hd, cod_max, precio_max);
  end;
end;
```

Solución Propuesta

```
{Variables del Programa Principal}
var
  p: propiedades;

begin
  p:= nil;
  generar_arbol(p);
  writeln('Código de propiedad con mayor cantidad de mts2: ', propiedad_mayor_mts(p));
  informar_direcciones_b(p, metros);
  writeln('Las propiedad que poseen entre ', limite_inferior,  ' y ', limite_superior, ' mts2 son: ');
  busqueda_acotada(p, limite_inferior, limite_superior);
  if (existe_propiedad(p, metros)) then
     writeln('Hay propiedades con exactamente ', metros, ' metros cuadrados.');
  writeln('La propiedad mas costosa es: ', propiedad_mas_costosa(p));
end.
```

¿Preguntas?