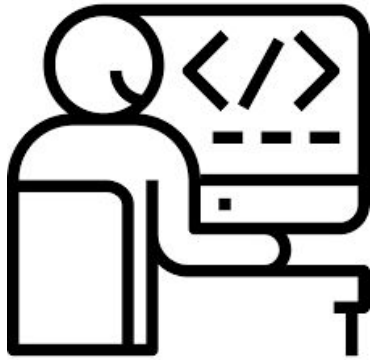


Algoritmos y Programación I

AyPI – **Temas de las clases pasadas**

- Tipos de Datos Arreglo
- Operaciones con vectores

AyPI – **Temas de la clase de hoy**

- Alocación estática y alocación dinámica
- Tipo de datos puntero
- Tipo de datos lista
- Características
- Operaciones

AyPI – **ALOCACIÓN ESTÁTICA y DINÁMICA** MEMORIA

MEMORIA ESTÁTICA

No se modifica su tamaño en tiempo de ejecución.

Las variables reservan memoria estática en su declaración. La suma de lo que ocupan se mantiene durante todo el programa. El lenguaje puede calcular el espacio estático requerido previo a la ejecución

MEMORIA DINÁMICA

Puede aumentar o disminuir el tamaño de la memoria utilizada en tiempo de ejecución.

AyPI – **TIPOS DE DATOS**

PUNTERO

SIMPLE: aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

COMPUESTO: pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.

- TIPO DE DATO
  - SIMPLE
    - DEFINIDO POR EL LENGUAJE
      - Integer
      - Real
      - Char
      - Boolean
      - Puntero
  - COMPUESTO
    - DEFINIDO POR EL LENGUAJE
      - String
    - DEFINIDO POR EL PROGRAMADOR
      - Registros
      - Arreglos

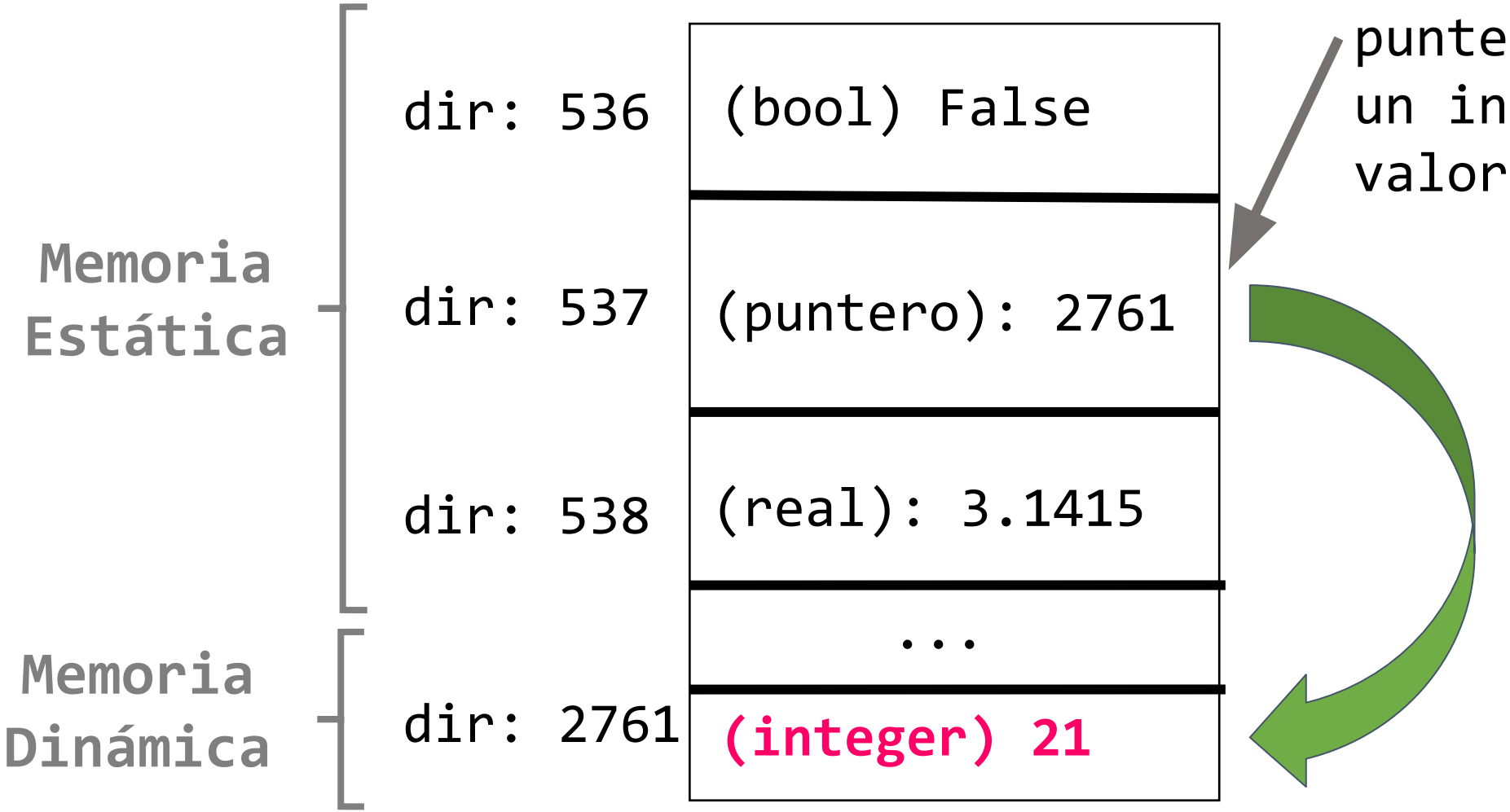
PUNTERO

Es un tipo de variable usada para almacenar una dirección en memoria. En esa dirección de memoria se encuentra el valor que puede ser de cualquiera de los tipos vistos (char, boolean, integer, real, string, registro, arreglo u otro puntero).
Un puntero es un tipo de datos simple.

AyPI – **TIPOS DE DATOS**

PUNTERO

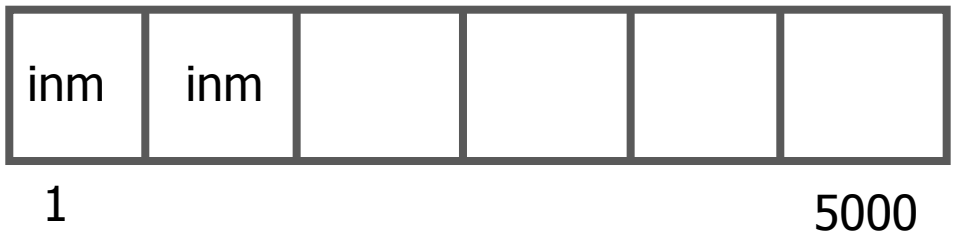
El contenido de la dirección 537 es un puntero que apunta a un integer (con el valor 21).



AyPI – **TIPOS DE DATOS**

¿Qué sucede si la inmobiliaria trabaja solo con 25 inmuebles?

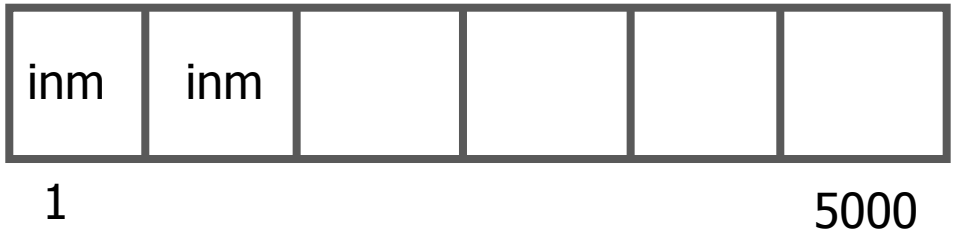
```
const
   CANT_INMUEBLES = 5000;
type
  inmueble = record
               ... end;
  vinmueble = array[1..CANT_INMUEBLES] of inmueble;

var
  vI: vinmueble;
```



¿Qué sucede si se debe trabajar con más de 5000 inmuebles?

AyPI – **TIPOS DE DATOS**

```
const
   CANT_INMUEBLES = 5000;
type
  inmueble = record
               ... end;
  vinmueble = array[1..CANT_INMUEBLES] of inmueble;

var
  vI: vinmueble;
```



¿Qué sucede si la inmobiliaria trabaja solo con 25 inmuebles?

Se desperdicia memoria que no se usa.

¿Qué sucede si se debe trabajar con más de 5000 inmuebles?

No puede. Habría que recompilar el programa luego de modificar la constante.

AyPI – **Tipo de dato Lista Enlazada**

SIMPLE: aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

COMPUESTO: pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.

- TIPO DE DATO
  - SIMPLE
    - DEFINIDO POR EL LENGUAJE
      - Integer
      - Real
      - Char
      - Boolean
      - Puntero
  - COMPUESTO
    - DEFINIDO POR EL LENGUAJE
      - String
    - DEFINIDO POR EL PROGRAMADOR
      - Registros
      - Arreglos
      - Lista

LISTA

Colección de nodos, donde cada nodo contiene un elemento y la dirección de memoria se encuentra el siguiente nodo.
Cada nodo de la lista se representa con un registro que contiene un dato y un puntero al siguiente nodo de la lista.

Los **nodos** que la componen pueden no ocupar posiciones contiguas de memoria. Es decir pueden aparecer dispersos en la memoria, pero mantienen un orden lógico interno.

Gráficamente …

AyPI – **Tipo de Dato**

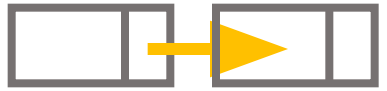
LISTA

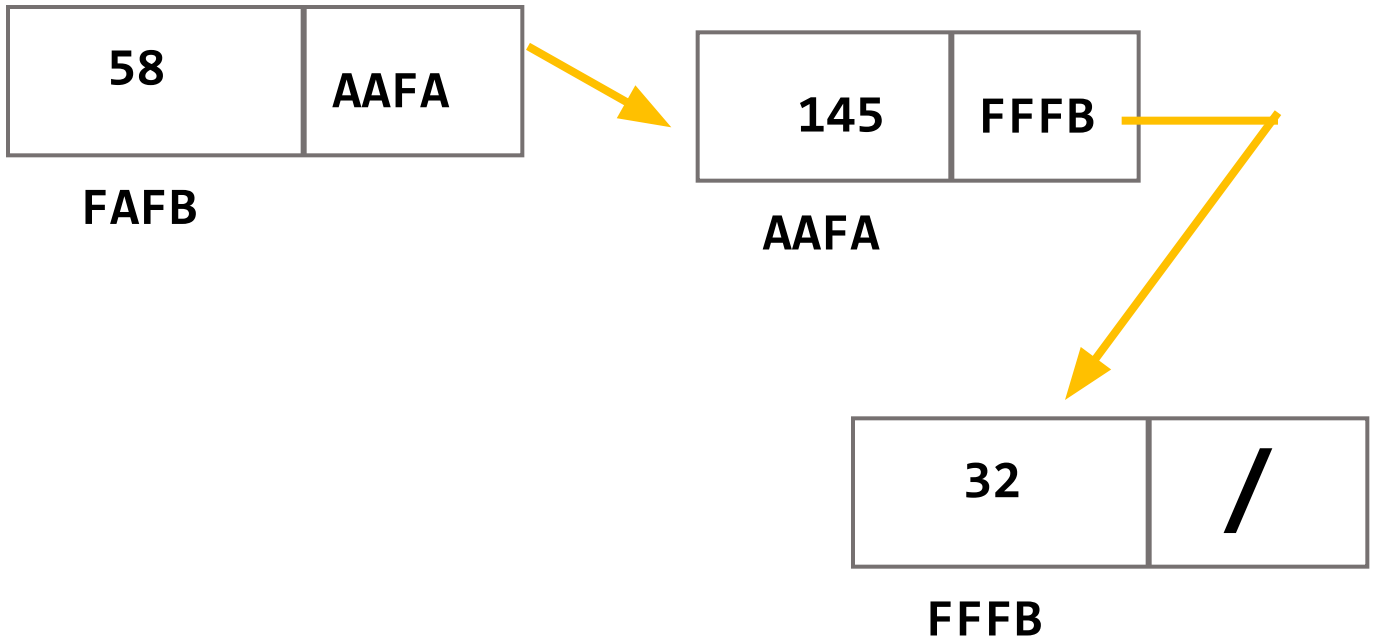
num: 536

pun: FAFB

145	FFFB	AAFA
58	AAFA	FAFB
32	nil	FFFB

LISTA





AyPI – **Tipo de Dato**

LISTA

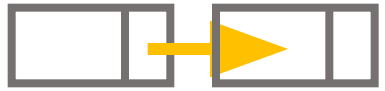
CARACTERÍSTICAS

Homogénea	Dinámica	Lineal	Secuencial
• Los elementos son todos del mismo tipo	• La cantidad de nodos puede variar durante la ejecución	• Cada nodo tiene un único antecesor y sucesor	• El acceso a cada elemento es de manera secuencial

¿Cómo se declara?

AyPI – **Tipo de Dato**

LISTA



```
program uno;

uses GenericLinkedList;

type
     Lista = specialize LinkedList<TIPO>;

var
  L: Lista;
```

Necesitamos incluir el tipo Lista y sus operaciones (módulos) (ya están programadas)

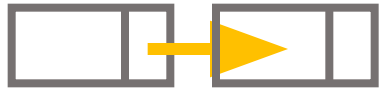
Cualquiera de los tipos vistos (incluidos los registros)

¿Lista de enteros?

¿Lista de registros?

AyPI – **Tipo de Dato**

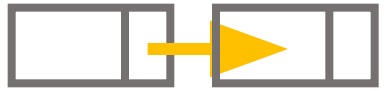
LISTA



```
program uno;
uses GenericLinkedList;

type
     ListaE = specialize LinkedList<integer>;

     inmueble = record
                  ...
                end;

     ListaI = specialize LinkedList<inmueble>;

var
  listaInmuebles: ListaI;
  listaEnteros: ListaE;
```

AyPI – **Tipo de Dato**

LISTA



OPERACIONES

Creación de una lista.

Agregar nodos al comienzo de la lista.

Recorrido de una lista.

Agregar nodos al final de la lista.

Insertar nodos en una lista ordenada

Eliminar nodos de una lista

Trabajaremos con una lista de enteros

AyPI – **Tipo de Dato - LISTA**

CREAR

CREAR UNA LISTA

Crea una lista vacía.

```
program uno;
uses GenericLinkedList;

type
    ListaE = specialize LinkedList<integer>;

var
    listaEnteros: ListaE;
begin

   listaEnteros := ListaE.create();

end.
```

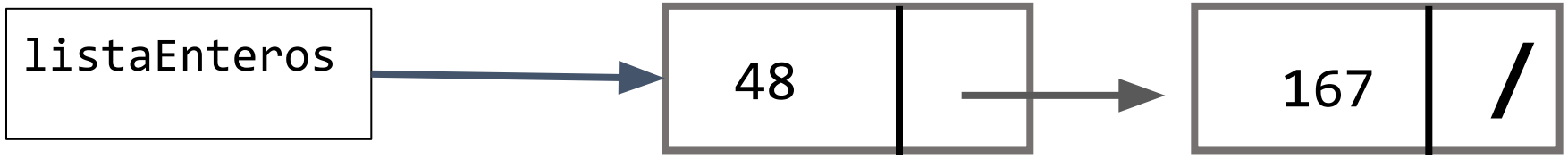
AGREGAR AL FINAL EN UNA LISTA

Implica generar un nuevo nodo y agregarlo como último elemento de la lista.

Agrego el 48

¿Cómo lo usamos en un ejemplo?

Agrego el 167



AGREGAR AL FINAL EN UNA LISTA

Implica invocar al modulo add de la lista.

```
Program uno;
uses GenericLinkedList;

Type
    ListaE = specialize LinkedList<integer>;
procedure armarLista(var L : listaE);
   …
Var
    listaEnteros: ListaE;
Begin
   armarLista(listaEnteros);
End.
```

```
procedure armarLista(var L : listaE);
var
  i, num: integer;
begin
    L := ListaE.create();
    for i:= 1 to 15 do begin
      read(num);
      L.add(num);
  end;
end;
```

RECORRER UNA LISTA

Implica posicionarse al comienzo de la lista y a partir de allí ir "pasando" por cada elemento de la misma hasta llegar al final.

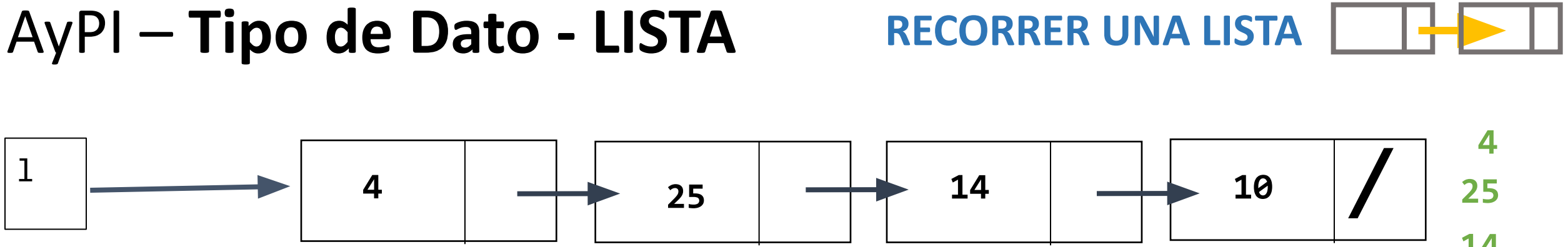
```
program uno;
uses GenericLinkedList;
type
    ListaE = specialize LinkedList<integer>;
     procedure armarLista(...)
     procedure recorrerLista( L : ListaE);
var
    listaEnteros: ListaE;
begin
  armarLista (listaEnteros);
  recorrerLista (listaEnteros);
end.
```

RECORRER UNA LISTA

```
Me posiciono en el comienzo de la lista

mientras (no sea el final de la lista)

    proceso el elemento actual (por ejemplo imprimo)
    avanzo al siguiente elemento
```

AyPI – **Tipo de Dato - LISTA**

RECORRER UNA LISTA



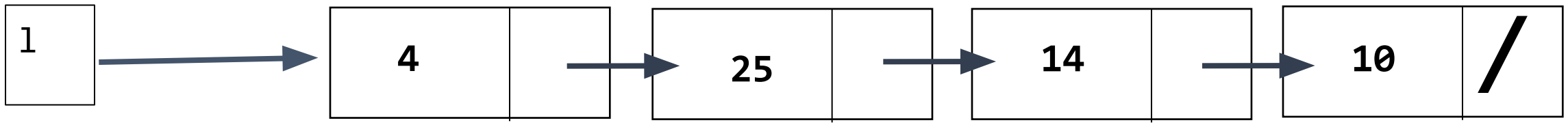

```
procedure recorrerLista (l: listaE);
begin
   l.reset();
   while (not l.eol()) do
    begin
        write (l.current());
        l.next();
    end;
end;
```

¿Funciona si la lista que recibo está vacía?

¿Qué cambios debo hacer si quiero que el procedimiento devuelva la suma de los elementos?

AyPI – **Tipo de Dato - LISTA**

RECORRER UNA LISTA




```
function SumarLista (l: listaE) : integer;
var
  suma : integer;
begin
  suma := 0;
  l.reset();
  while (not l.eol()) do
   begin
      suma := suma + l.current();
      l.next();
   end;
   SumarLista := suma;
end;
```

Una inmobiliaria nos encargó un programa para el procesamiento de sus inmuebles. De cada inmueble se deberán considerar las siguientes características: código de identificación del inmueble, tipo, cantidad de habitaciones, cantidad de baños, precio, localidad y fecha de publicación.
Implementar módulos para cada uno de los siguientes ítems:

a. Leer la información de los inmuebles hasta que se ingresa el código -1, y almacenarlos en una lista.

b. Informar todos los códigos de los inmuebles que tienen más habitaciones que una cantidad que se recibe como parámetro (debe ser leída en el programa principal).

c. Retornar los inmuebles agrupados para cada mes de publicación.

d. Informar los códigos de los inmuebles para un mes que se recibe como parámetro utilizando la información generada en c) (el mes debe ser leído en el programa principal).