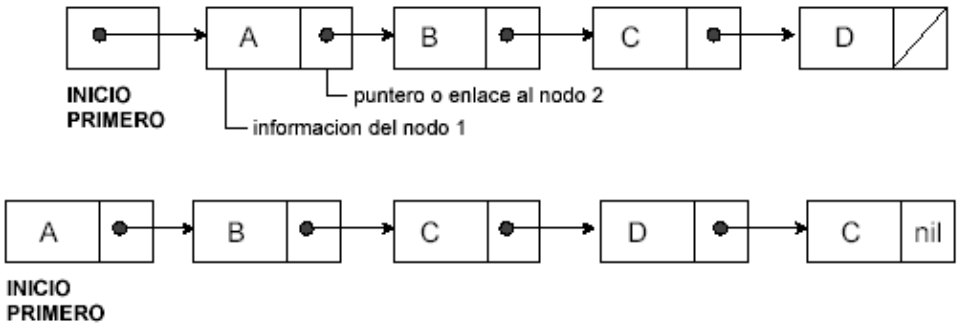
Las estructuras dinámicas son una implementación de TDAs o TADs (Tipos Abstractos de Datos). En estos tipos el interés se centra más en la estructura de los datos que en el tipo concreto de información que almacenan.
Dependiendo del número de punteros y de las relaciones entre nodos, podemos distinguir varios tipos de estructuras dinámicas.

Las listas enlazadas están formadas por un conjunto de nodos, en los que cada elemento contiene un puntero con la posición o dirección del siguiente elemento de la lista, es decir su enlace. Se dice entonces, que los elementos de una lista están enlazados por medio de los campos enlaces. Cada nodo está constituido por dos partes: información o campos de datos (uno o varios) y un puntero (con la dirección del nodo siguiente). Al campo o campos de datos del nodo lo designaremos como INFORMACIÓN del nodo y al puntero por SIGUIENTE del nodo. Con esta organización de los datos, es evidente que no es necesario que los elementos de la lista estén almacenados en posiciones físicas adyacentes para estar relacionados entre sí, ya que el puntero indica unívocamente la posición del dato siguiente en la lista.
Nótese que para tener definida una lista enlazada, además de la estructura de cada uno de sus nodos, necesitamos una variable externa a la propia lista, con un puntero que marque la posición de la cabeza (inicio, primero) de la lista. Esta variable es quien normalmente da nombre a la lista, pues nos dice donde localizarla, sea cual sea su tamaño. Para detectar el último elemento de la lista se emplea un puntero nulo, que por convenio, se suele representar de diversas formas: por un enlace con la palabra reservada nil (NULO), por una barra inclinada (/) o por un signo especial, tomado de la toma de tierra en electricidad. Una lista enlazada sin ningún elemento se llama lista vacía y las distinguiremos asignando a su puntero de cabecera el valor nil.



El nodo típico para construir listas tiene esta forma:

```
struct nodo {
   int informacion;
   struct nodo *siguiente;
};
```

En el ejemplo, cada elemento de la lista sólo contiene una informacion de tipo entero, pero en la práctica no hay límite en cuanto a la complejidad de los datos a almacenar.

Normalmente se definen varios tipos que facilitan el manejo de las listas, en C, la declaración de tipos puede como esta esta:

```
typedef struct _nodo {
   int dato;
   struct _nodo *siguiente;
} tipoNodo;
```

```
typedef tipoNodo *pNodo;
typedef tipoNodo *Lista;
```

Donde **tipoNodo** es el tipo para declarar nodos, evidentemente y **pNodo** es el tipo para declarar punteros a un nodo.
Lista es el tipo para declarar listas, como puede verse, un puntero a un nodo y una lista son la misma cosa. En realidad, cualquier puntero a un nodo es una lista, cuyo primer elemento es el nodo apuntado.

En nuestro proyecto este es el ejemplo de una declaración de una de las listas:

```
typedef struct NodoCliente {
        char cedula[12];
        char nombre[30];
        char apellido[30];
        char procedencia[30];
        char telefono[10];

        struct NodoCliente *sigC;
}Cliente;
typedef Cliente *CNodo;
typedef Cliente *CLista;
```

Es muy importante que nuestro programa nunca pierda el valor del puntero al primer elemento, ya que si no existe ninguna copia de ese valor, y se pierde, será imposible acceder al nodo y no podremos liberar el espacio de memoria que ocupa.

ENUNCIADO DEL PROGRAMA

Hotel Carlanchin necesita un programa

- En una lista A, almacena los datos de cada cliente:
  - Nº de cedula
  - Nombre
  - Apellido
  - Procedencia
  - Teléfono.
- En una lista B los datos de cada habitación:
  - Nº de habitación
  - Piso
  - Tipo
  - Valor
- En la lista C se almacena la facturación y consumo.

Diseñe un programa que nos permita realizar búsquedas por cliente, búsquedas por habitación e imprima facturación por cliente.

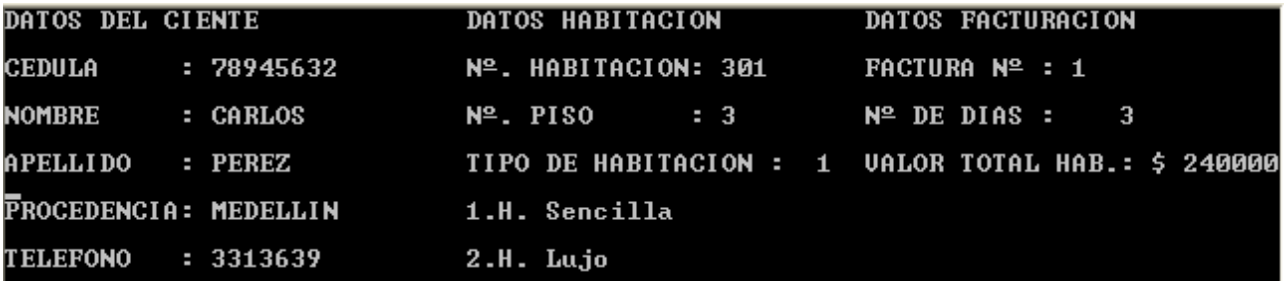
DATOS DEL PROGRAMA

Nombre del programa: **Hotel Carlanchin**
Nombre del archivo fuente: **hotel.CPP**
Nombre del archivo ejecutable: **hotel.EXE**
Fecha de creación: **11-05-2005**
Compilador usado: **TURBO C++ IDE 3.0**

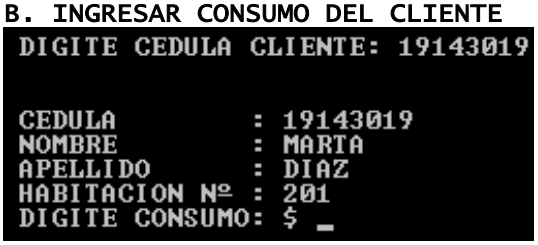
IMPRESIÓN DE PANTALLA DE RESULTADO

MENU

A. INGRESAR DATOS



B. INGRESAR CONSUMO DEL CLIENTE



C. IMPRIMIR FACTURAS

D. BUSQUEDA POR HABITACION

```
HABITACION Nº          : 201

PISO                   : 2
NOMBRE                 : MARTA
APELLIDO               : DIAZ
CEDULA                 : 19143019
TIPO                   : HABITACION DE LUJO
VALOR HABITACION DIA   : $ 100000
Nº DIAS                : 3
CONSUMO                : $ 60000
VALOR TOTAL            : $ 360000
```

E. BUSQUEDA POR CLIENTE

```
HABITACION Nº : 301
CEDULA        : 78945632
NOMBRE        : CARLOS
APELLIDO      : PEREZ
PROCEDENCIA   : MEDELLIN
TELEFONO      : 3313639
/-------------------------------------/
```

F. MOSTRAR LISTA CLIENTES

CEDULA	NOMBRE	APELLIDO	PROCEDENCIA	TELEFONO
19143019	MARTA	DIAZ	CARTAGENA	5698653
78945632	CARLOS	PEREZ	MEDELLIN	3313639