Universidad Nacional de Ingeniería

Facultad de Ciencias

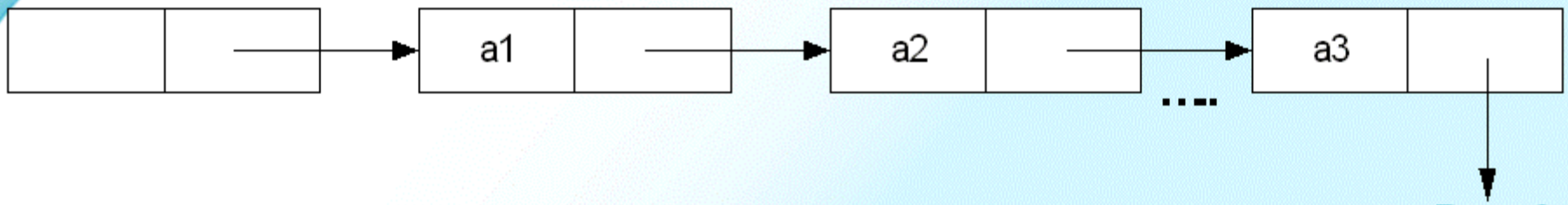
Introducción a la Programación (CC-102)

Sesión 11: Listas Enlazadas - 1

DEFINICIÓN

Una **lista enlazada** es una colección o secuencia de elementos dispuestos uno detrás de otro, en la que cada elemento se conecta al siguiente elemento por un «enlace» o «puntero».

Los elementos de una lista se llaman **nodos** y se componen de al menos dos partes o *campos:* la primera parte o campo contiene la información y la segunda parte o *campo* es un puntero (denominado *enlace* o *sig)* que apunta al siguiente elemento de la lista.



CARACTERISTICAS

- Los nodos de una lista enlazada no ocupan posiciones contiguas en memoria.
- El tamaño de la estructura puede aumentar y disminuir durante la ejecución del programa.
- Si la lista está vacía, comienzo es **NULL**.
- La lista se considera llena cuando no existe espacio disponible para crear una variable dinámica de tipo nodo.
- No todos los lenguajes de programación permiten la implementación de las listas enlazadas dinámicas

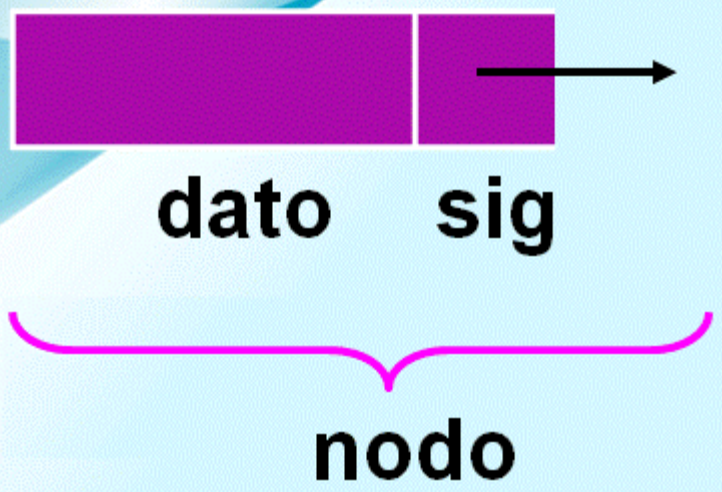
CLASIFICACIÓN DE LAS LISTAS ENLAZADAS

Simplemente enlazadas	Doblemente enlazadas	Circular simplemente enlazada	Circular doblemente enlazada
Cada nodo (elemento) contiene un único enlace que conecta ese nodo al nodo siguiente o nodo sucesor. La lista es eficiente en recorridos directos ((<adelante»).	Cada nodo contiene dos enlaces, uno a su nodo predecesor y el otro a su nodo sucesor. La lista es eficiente tanto en recorrido directo («adelante») como en recorrido inverso («atrás»).	Una lista enlazada simplemente en la que el último elemento (cola) se enlaza al primer elemento (cabeza) de tal modo que la lista puede ser recorrida de modo circular («en anillo»).	Una lista doblemente enlazada en la **que** el último elemento se enlaza al primer elemento y viceversa. Esta lista se puede recorrer de modo circular (en anillo) tanto en dirección directa («adelante») como inversa («atrás»).

Por cada uno de estos cuatro tipos de estructuras de listas, se puede elegir una implementación basada en arrays (asignación fija o estática) o una implementación basada en punteros (asignación dinámica de memoria mediante punteros).

OPERACIÓN: DECLARAR UN NODO

```
C
struct nodo{
        int dato;
        struct nodo *sig;
};
```



EJERCICIO 1

Elabore un programa que cree dinámicamente una lista enlazada la que almacenará el nombre y la nota de alumnos de un salón de clase.

DECLARACIÓN DE CABECERAS Y LISTA

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Lista{
   char nombre[41];
   int  nota;
   struct Lista *sig;
};
```

DECLARACIÓN DE VARIABLES Y RESERVA DE MEMORIA

```
int main(void)
{
   typedef struct Lista t_Lista;
   t_Lista *nuevo,*actual;
   t_Lista *inicio;
   int i=0,num,resp;
   char cad[80],alu[80];

   nuevo=malloc(sizeof(t_Lista));
   if (nuevo==NULL){
      printf("Error de asignacion de memoria");
      exit(-1);
   }
   nuevo->sig=NULL;
```

INSERCIÓN EN LA LISTA

```
inicio=nuevo;
   do{
        printf("alumno %d:",i+1);
        gets(alu);
        strcpy(nuevo->nombre,alu);

        printf("nota: ");
        scanf("%d",&num);
        getchar();
        nuevo->nota=num;

        nuevo->sig=malloc(sizeof(t_Lista));
        if (nuevo->sig==NULL){
              printf("Error de asignacion de memoria");
              exit(-1);
        }
        nuevo=nuevo->sig;
        nuevo->sig=NULL;
        printf("Continuar(1=Si/0=No)?: ");
        scanf("%d",&resp);
        getchar();
        i++;
   } while (resp==1);
```

RECORRER LA LISTA ENLAZADA

```
actual=inicio;
while (actual->sig!=NULL){
    printf("\nAlumno=%s,Nota=%d",actual->nombre,actual->nota);
    actual=actual->sig;
}
printf("\n");
```

LIBERAR DE MEMORIA LA LISTA ENLAZADA

```
    nuevo=inicio;
    while (nuevo->sig!=NULL){
      t_Lista *sig;
      sig=nuevo->sig;
      free(nuevo);
      nuevo=NULL;
      nuevo=sig;
    }
    return  0;
}//main
```