Insertar y eliminar nodos en una lista simplemente enlazada

Introducción

En este módulo, conoceremos en detalle cómo se insertan y se eliminan nodos en una lista. Esto se realiza durante la ejecución del programa, por ello las listas son estructuras dinámicas: se reserva memoria para insertar datos y se libera la memoria para eliminarlos.

Inserción

Para insertar un nodo en una lista, hay que seguir lo siguientes pasos:

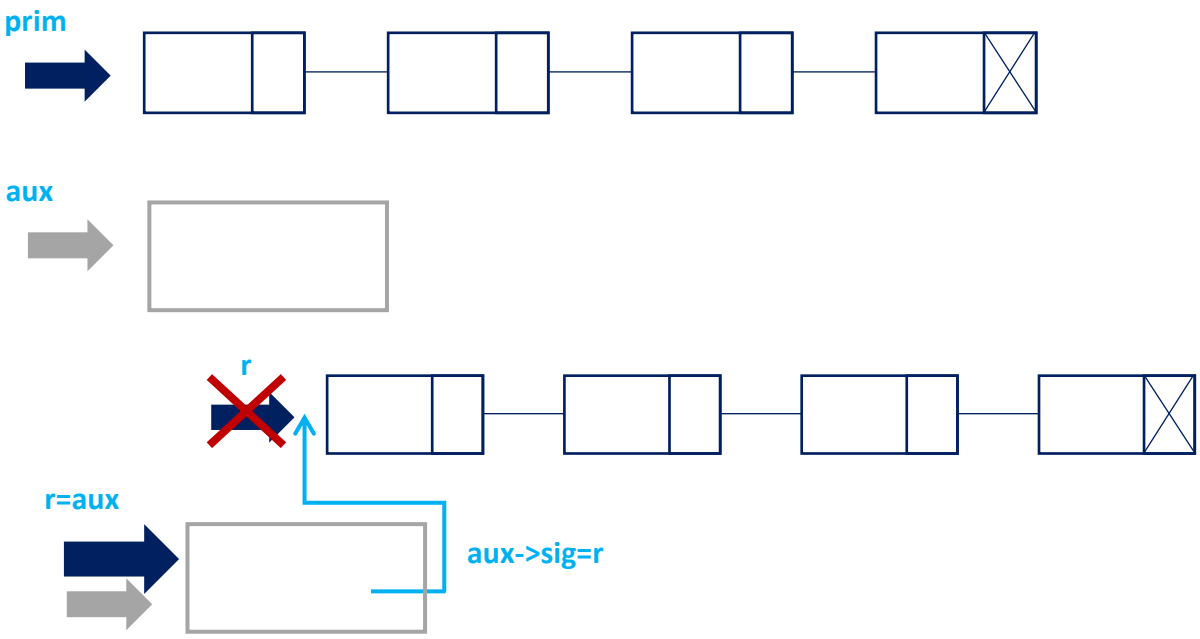
- Reservar espacio de memoria (con la función Malloc; la cantidad se calcula con la función Sizeof).
- Enlazar el nuevo nodo con la lista.
- Enlazar la lista con el nuevo nodo.

Hay que distinguir dos tipos de inserción: por cabeza de lista, es decir, insertar un nodo al comienzo de la lista y en cualquier lugar de la lista.

Inserción de un nodo por cabeza de lista

Para insertar un nodo por cabeza de lista es necesario construir una función que permita devolver la dirección del puntero a cabeza de lista que se ha modificado, para indicárselo al programa principal.

En la siguiente lista (solo como ejemplo) se detallan los enlaces necesarios para la inserción del nuevo nodo. El puntero a cabeza de lista es prim, pero al llamar a la función insertar por cabeza de lista, esa dirección la recibe el puntero que denominaremos r.



Aquí están detallados los pasos:

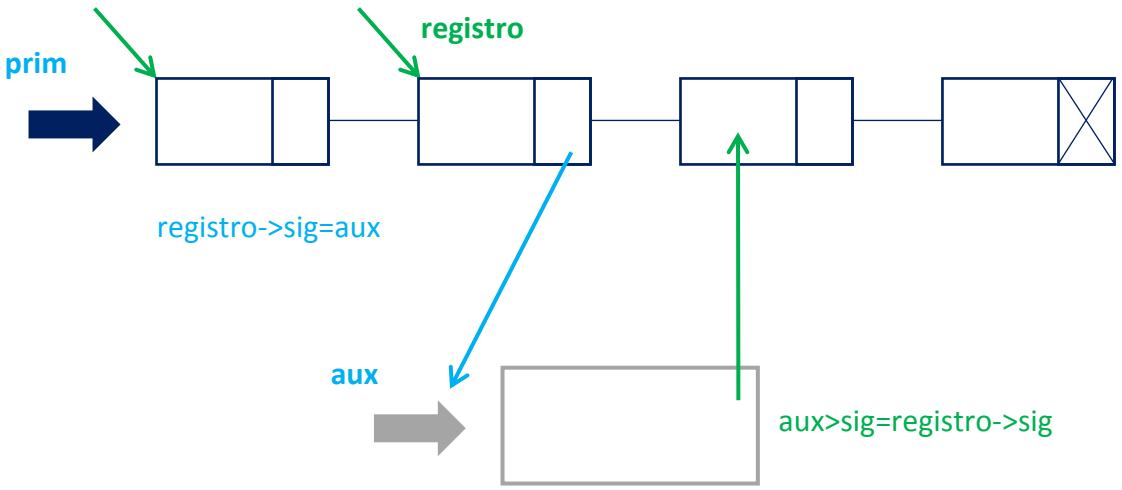
1. Se reserva memoria y esa dirección la toma el puntero aux.
2. Se realiza el primer enlace del nuevo nodo a la lista.
3. Se modifica el puntero a cabeza de lista (r) para que apunte a la dirección de aux.
4. Se devuelve esta dirección al main, que es recibida por el puntero a cabeza de lista (prim) que se actualiza.

Código de la función insertar por cabeza de lista

```
 nodo* insertar1(nodo *r)
{
    nodo *aux;
       if()//condicion de insercion
    {
        aux=(nodo *)malloc(sizeof(nodo));
        aux->num=0;//valor a insertar
        aux->sig=r;
        r=aux;
    }
    return r;
}
```

Inserción de un nodo en cualquier lugar de la de lista

En este caso, no se modifica la dirección del puntero a cabeza de lista.



Aquí están detallados los pasos:

1. Se reserva memoria y esa dirección la toma el puntero aux.
2. Se realiza el primer enlace del nuevo nodo a la lista.
3. Se enlaza la lista al nuevo nodo.

Código de la función insertar en cualquier lugar de la lista

```
void insertar2 (nodo *registro)
{nodo *aux=NULL;
    while(p->sig!=NULL)
    {
        if() //condicion de insercion
        {
            aux=(nodo *)malloc(sizeof(nodo));
            aux->num=0;//valor a insertar
            aux->sig=registro->sig;
            registro->sig=aux;

        }
        registro=registro->sig;
    }
}
```

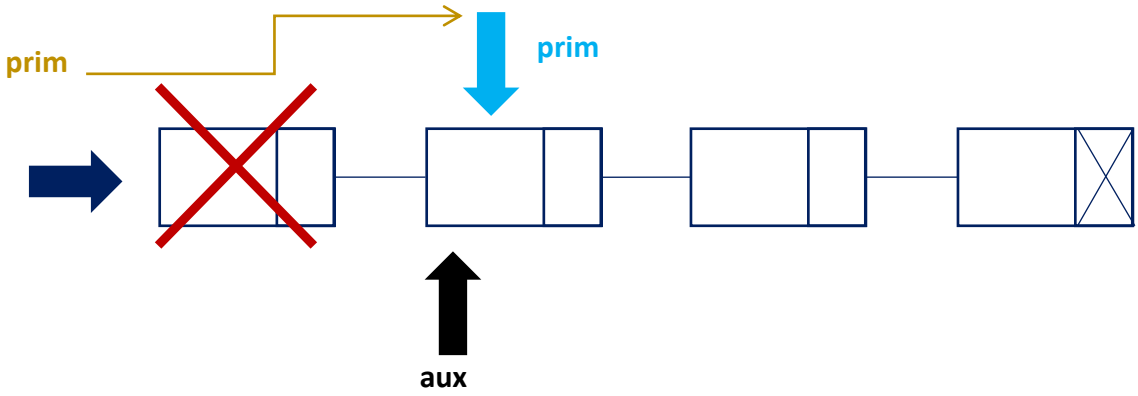
Eliminación de nodos en una lista

Para eliminar un nodo en una lista hay que seguir lo siguientes pasos:

- Se establece un puntero al siguiente nodo que se quiere eliminar.
- Se libera el espacio de memoria reservado con la función Malloc con la función Free.
- Enlazar los nodos de la lista.

Eliminación de un nodo por cabeza de lista

Al eliminar nodos por cabeza de lista, se debe actualizar la dirección del puntero a cabeza de lista en el programa principal. Para esto, se debe construir una función que devuelva esta dirección.
Gráficamente:



Enlace y liberación de la memora:

- aux=prim->sig
- free(prim)
- prim=aux

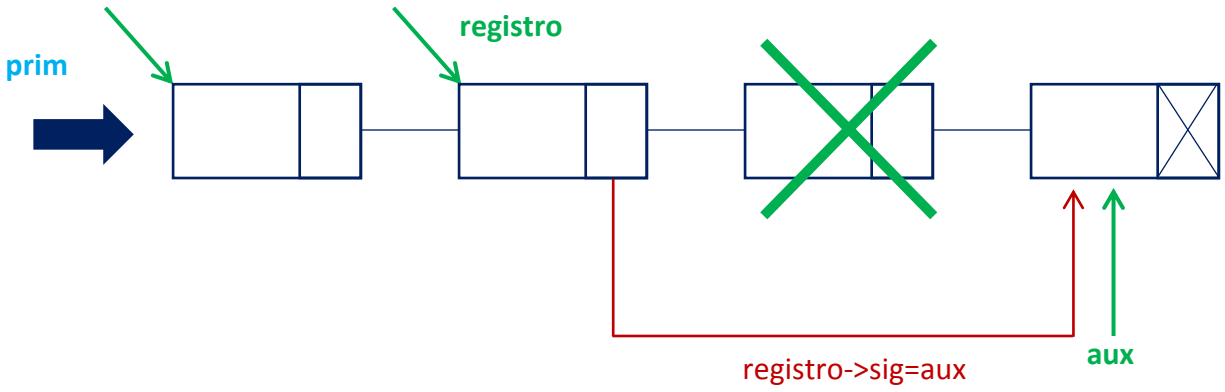
Código de la función eliminar por cabeza de lista

```
nodo* eliminar1(nodo* prim,)
{nodo *aux;
    while()//condicion de eliminacion
    {
        aux=prim->sig;
        free(prim);
        prim=aux;

    }
    return prim;
}
```

Eliminar un nodo en cualquier lugar de la de lista

En este caso, no se modifica la dirección del puntero a cabeza de lista.
Gráficamente:



Cuando se borra un nodo no se avanza en la lista ya que no se podrían borrar nodos consecutivos, cuando no se borra si se avanza.

Código de la función eliminar en cualquier lugar de la lista

```
void eliminar2(nodo* registro)
{nodo *aux=NULL;
    while(registro->sig!=NULL&& registro->sig->sig!=NULL)
    {
        if()//condicion de eliminacion
        {
            aux=registro->sig->sig;
            free(registro->sig);
            registro->sig=aux;
        }
        else
        {
            registro=registro->sig;
        }
    }

}
```