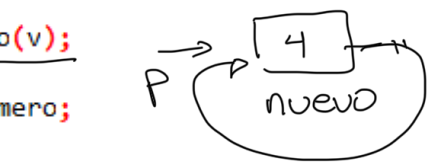
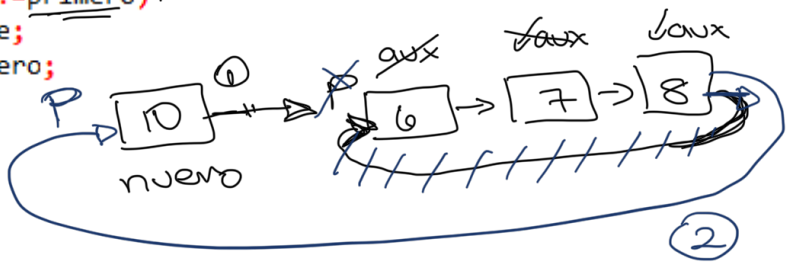
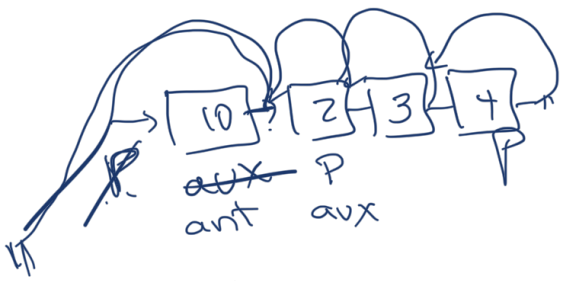
```
void listaC::InsertarInicio(int v) //4->2-9
{
    if (ListaVacia())
    {
      pnodo nuevo= new nodo(v);
      primero = nuevo;
      nuevo->siguiente=primero;
    }
    else
    {
      pnodo nuevo= new nodo(v);
      pnodo aux=primero;
      while (aux->siguiente!=primero)
         aux= aux->siguiente;
      nuevo->siguiente=primero;
      aux->siguiente=nuevo;
      primero=nuevo;
    }
}
```





10 -> 2 -> 3 -> 4 || 4 -> 3 -> 2 -> 10



Construir en el main una lista

L1. II (10);
L1. IF (2);
L. Mostrar
L. Invertir

ant = NULL

aux -> sig = ant

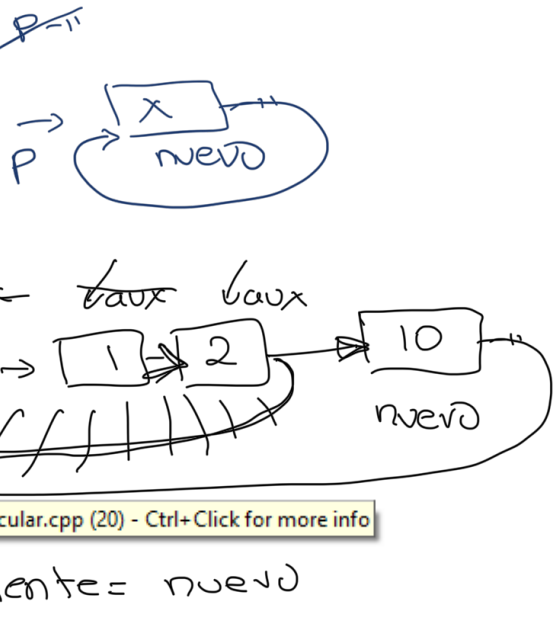
L1. Invertir()
L1. Mostrar

punteros

1. No crear una lista nueva
2. No mover los valores

```
void listaC::InsertarFinal(int v)
{
   if (ListaVacia())
   {
     pnodo nuevo= new nodo(v);
     primero = nuevo;
     nuevo->siguiente=primero;
   }
   else
   {
     pnodo nuevo=new nodo(v);
     pnodo aux = primero;
     while (aux->siguiente!=primero)
        aux= aux->siguiente;
   ->nuevo->siguiente= primero;
     aux->siguient
   }
}
```

private nodo* nodo::siguiente - ListaCircular.cpp (20) - Ctrl+Click for more info



aux->siguiente= nuevo

```
void listaC::InsertarPos(int v,int pos)
{
   if (ListaVacia())
   {
     pnodo nuevo= new nodo(v);
     primero = nuevo;
     nuevo->siguiente=primero;
   }
   else
   {
     if(pos <=1)
     {
       InsertarInicio(v);
     }
     else
     {
       pnodo aux= primero;
       int i =2;
       while((i != pos )&&(aux->siguiente!= primero))
       {
          i++;
          aux=aux->siguiente;
       }
       pnodo nuevo= new nodo(v);
       nuevo->siguiente=aux->siguiente;
       aux->siguiente=nuevo;

      }
     }
}
```

```
void listaC::BorrarFinal()
{
    if (ListaVacia())
      cout << "No hay elementos en la lista:" << endl;
    else
    {
      if (primero->siguiente == primero)
      {
        pnodo temp= primero;
        primero= NULL;
        delete temp;
      }
      else
      {
        pnodo aux = primero;
        while (aux->siguiente->siguiente != primero)
            aux = aux->siguiente;
        pnodo temp = aux->siguiente;
        aux->siguiente= primero;
        delete temp;
      }
    }
}
```