**ÚLTIMA ACTUALIZACIÓN: 16/05/2024 14:36**

[**prueba\_1**](https://pythontutor.com/render.html#code=%23include%20%3Cstdio.h%3E%0A%23include%20%3Cstdlib.h%3E%0Astruct%20s_contenido%7B%0A%20%20%20%20int%20valor%3B%20%20%20%20%0A%7D%3B%0Atypedef%20struct%20s_contenido%20t_contenido%3B%0Astruct%20s_nodo%7B%0A%20%20%20%20t_contenido%20content%3B%0A%20%20%20%20struct%20s_nodo*%20sig%3B%0A%7D%3B%0Atypedef%20struct%20s_nodo*%20t_nodo%3B%0A%0Avoid%20insertarEnListaUltimo%28t_nodo*%20nodo,%20t_contenido%20carga%29%7B%0A%20%20%20%20if%20%28%20*nodo%20%3D%3D%20NULL%29%7B%0A%20%20%20%20%20%20%20%20//%20crear%20el%20nodo%0A%20%20%20%20%20%20%20%20*nodo%20%3D%20%20malloc%28sizeof%28struct%20s_nodo%29%29%3B%0A%20%20%20%20%20%20%20%20%28*nodo%29-%3Econtent%3Dcarga%3B%20%20%20%20%20//%20cargar%20el%20contenido%0A%20%20%20%20%20%20%20%20%28*nodo%29-%3Esig%3DNULL%3B%20%20%20%20%20%20%20%20%20%20//colocar%20null%20al%20puntero%0A%20%20%20%20%20%20%20%20%0A%20%20%20%20%7Delse%7B%0A%20%20%20%20%20%20%20%20//%20llamada%20recursiva%0A%20%20%20%20%20%20%20%20insertarEnListaUltimo%28%26%28%28*nodo%29-%3Esig%29,%20carga%29%3B%20%20//%20%20%20%20%20%20%20%20%20%0A%20%20%20%20%7D%0A%7D%0A%0Avoid%20imprimirListaR%28t_nodo%20lista%29%7B%0A%20%20%20%20%0A%20%20%20%20%0A%7D%0Aint%20main%28%29%20%7B%0A%20%20%20%20t_nodo%20lista%20%3D%20NULL%3B%0A%20%20%20%20t_contenido%20aux%3B%0A%20%20%20%20aux.valor%3D10%3B%0A%20%20%20%20insertarEnListaUltimo%28%26lista,%20aux%29%3B%0A%20%20%20%20aux.valor%3D20%3B%0A%20%20%20%20insertarEnListaUltimo%28%26lista,%20aux%29%3B%0A%20%20%20%20aux.valor%3D30%3B%0A%20%20%20%20insertarEnListaUltimo%28%26lista,%20aux%29%3B%0A%20%20%20%20%0A%20%20%20%20imprimirListaR%28lista%29%3B%20%20%20%20%0A%20%20%20%20return%200%3B%0A%7D&cumulative=false&curInstr=49&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D&textReferences=false)

[**prueba\_2**](https://pythontutor.com/visualize.html#code=%23include%20%3Cstdio.h%3E%0A%23include%20%3Cstdlib.h%3E%0Astruct%20s_nodo%20%7B%0A%20%20%20%20int%20val%3B%0A%20%20%20%20struct%20s_nodo*%20sig%3B%0A%7D%3B%0Atypedef%20struct%20s_nodo*%20t_nodo%3B%0A%20%20%20%20//%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2380%20%20%20%20%2020%0Avoid%20append%28t_nodo*%20nodo,int%20val%29%7B//%20agregar%20a%20lo%20ultimo%0A%20%20%20%20if%20%28%20*nodo%20%3D%3D%20NULL%29%7B%0A%20%20%20%20%20%20%20%20//%20crear%20el%20nodo%0A%20%20%20%20%20%20%20%20*nodo%20%3D%20%20malloc%28sizeof%28struct%20s_nodo%29%29%3B%0A%20%20%20%20%20%20%20%20%28*nodo%29-%3Eval%3Dval%3B%20%20%20%20%20%20%20%20%20%20%20//%20cargar%20el%20contenido%0A%20%20%20%20%20%20%20%20%28*nodo%29-%3Esig%3DNULL%3B%20%20%20%20%20%20%20%20%20%20//colocar%20null%20al%20puntero%0A%20%20%20%20%7D%20%20%20%20%0A%20%20%20%20else%7B%0A%20%20%20%20%20%20%20%20append%28%20%26%28%28*nodo%29-%3Esig%29,%20val%29%3B%0A%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%0A%7D%0Avoid%20imprimirListaR%28t_nodo%20ls%29%7B%0A%20%20%20%20//%20Imprimir%20Recursivo%0A%20%20%20%20if%28ls!%3DNULL%29%7B%0A%20%20%20%20%20%20%20%20printf%28%22%5Cn%20valor%3D%254d,%20dirNodo%3A%3C%25p%3E,%20dirSig%3A%3C%25p%3E%22,%28ls%29-%3Eval,ls,ls-%3Esig%29%3B%0A%20%20%20%20%20%20%20%20imprimirListaR%28ls-%3Esig%29%3B%0A%20%20%20%20%7D%0A%7D%0A%0Aint%20main%28int%20argc,%20char%20*argv%5B%5D%29%20%7B%0A%20%20%20%20t_nodo%20ls%3DNULL%3B//%23110%20%28direcc%20de%20ls%20-%3E%20%2380%29%0A%20%20%20%20append%28%26ls,%2010%29%3B%0A%20%20%20%20append%28%26ls,%2020%29%3B%0A%20%20%20%20append%28%26ls,%2030%29%3B%0A%20%20%20%20imprimirListaR%28ls%29%3B%0A%20%20%20%20return%200%3B%0A%7D&cumulative=false&heapPrimitives=nevernest&mode=edit&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D&textReferences=false)

**prueba\_3**

| **#include <stdio.h> #include <stdlib.h>  */\*\*  L I S T A \*/*  */\*typedef struct sNodo{\*/* */\* int valor;\*/* */\* struct sNodo \*sig;\*/* */\*}\*tNodo;\*/* */\*-----------------------------\*/*  struct sNodo{  int valor;  struct sNodo\* sig; };**  **typedef struct sNodo\* tNodo;  */\*----------------------------------------------\*/*  void insertarEnListaPrimero(tNodo\*, int); void insertarEnListaUltimo(tNodo\*, int); void insertarEnListaPorPosicion(tNodo\*, int, int); void insertarEnListaOrdenado(tNodo\*, int);  void eliminarEnListaPorValor(tNodo\* nodo, int ); int eliminarEnListaPorPosicion(tNodo\* nodo, int );  void imprimirListaI(tNodo); *// Iterativo* void imprimirListaR(tNodo); *// Recursivo* void imprimirListaInversaR(tNodo); *// Recursivo***  **int largoLista(tNodo); void eliminarLista(tNodo\*); int esListaVacia(tNodo);**  ***/\*----------------------------------------------\*/***  **int main() {   */\*--------------------\*/*  unsigned int n,c;  tNodo lista = NULL;  printf("\n L I S T A S \n");  */\*--------------------------------------------------\*/*  printf("\n==================================================\n");  printf("Agrega valores a lista [10,20,30]");  printf("\n--------------------------------------------------\n");   insertarEnListaUltimo(&lista, 10);  insertarEnListaUltimo(&lista, 20);  insertarEnListaUltimo(&lista, 30);  imprimirListaR(lista);  */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("inserta adelante el 114 y luego el 7 \n");  printf("resultando la lista [7,114,10,20,30]");  printf("\n--------------------------------------------------\n");   insertarEnListaPrimero(&lista,114);  insertarEnListaPrimero(&lista,7);  imprimirListaR(lista);    */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("De la lista anterior \n");  printf("se elimina el valor 114\n");  printf("resultando la lista [7,10,20,30]");  printf("\n--------------------------------------------------\n");   eliminarEnListaPorValor(&lista,114);  imprimirListaR(lista);      */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("inserta ordenado el 22 y el 3 en la lista \n");  printf("resultando la lista [3,7,10,20,22,30]");  printf("\n--------------------------------------------------\n");   insertarEnListaOrdenado(&lista, 22);  insertarEnListaOrdenado(&lista, 3);   imprimirListaI(lista);      */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("De la lista anterior \n");  printf("Se elimina la posición 1");  printf("\n--------------------------------------------------\n");    printf("-- Se elimino la posicion 1 y el valor \n");  int val=eliminarEnListaPorPosicion(&lista, 1);  printf("-- extraido es%d\n-- y se vuelve imprimir\n ",val);    imprimirListaI(lista);  */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("De la lista anterior \n");  printf("Se inserta en posicion 1 el valor 111");  printf("\n--------------------------------------------------\n");  printf("\n-- insertando ---\n");  insertarEnListaPorPosicion(&lista,1,111);   imprimirListaI(lista);    */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("La lista anterior \n");  printf("la imprime invertida");  printf("\n--------------------------------------------------\n");   imprimirListaInversaR(lista);    */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf(" Cuenta los elemento de la lista 'como el len' \n");  printf(" Elimina la lista, ojo elimina los elementos \n");  printf(" Verifica si la lista está vacia");  printf("\n--------------------------------------------------\n");   printf("Largo de la lista = %d",largoLista(lista));  printf("\n... Eliminando lista ...\n");  eliminarLista(&lista);  if (esListaVacia(lista))  printf(" :) La lista esta VACIA\n");  else  printf(" :( ERROR La lista AUN esta LLENA \n");  printf("Largo de la lista = %d",largoLista(lista));     return 0; }**  **void insertarEnListaUltimo(tNodo\* nodo, int valor){  *//insertar <al final de la lista>*  if ( \*nodo == NULL){  \*nodo = (tNodo) malloc(sizeof(struct sNodo));  (\*nodo)->valor = valor; *// (\*(\*nodo)).valor = valor;*  (\*nodo)->sig = NULL;  }  else{  insertarEnListaUltimo(&((\*nodo)->sig), valor);    } }**  **void insertarEnListaPorPosicion(tNodo\* nodo, int pos, int valor){  if ( pos==0 || \*nodo == NULL){   tNodo aux;  if ( \*nodo == NULL) {  \*nodo = (tNodo) malloc(sizeof(struct sNodo));  (\*nodo)->valor = valor;  (\*nodo)->sig = NULL;  }  else {  aux = (tNodo) malloc(sizeof(struct sNodo));  aux->sig=(\*nodo);  aux->valor =valor ;  (\*nodo)=aux;  }   }  else if(pos>0){  insertarEnListaPorPosicion(&((\*nodo)->sig), pos-1,valor);  }  }     void insertarEnListaOrdenado(tNodo\* nodo, int valor){  tNodo aux=NULL;  if ( \*nodo == NULL){ */\*\*INSERTO ULTIMO\*/*  \*nodo = (tNodo) malloc(sizeof(struct sNodo));  (\*nodo)->valor = valor;  (\*nodo)->sig = NULL;  }  else {  if ((\*nodo)->valor > valor){  aux = (tNodo) malloc(sizeof(struct sNodo));  aux->sig=(\*nodo);  aux->valor =valor ;  (\*nodo)=aux;  }  else{  insertarEnListaOrdenado(&(\*nodo)->sig, valor);  }  } }  void insertarEnListaPrimero(tNodo\* nodo, int valor) {  tNodo aux;  aux = (tNodo) malloc(sizeof(struct sNodo));  aux->sig=(\*nodo);  aux->valor =valor ;  (\*nodo)=aux; }**  **void eliminarEnListaPorValor(tNodo\* nodo, int valor) {  tNodo aux = NULL;  if ( \*nodo != NULL) {  if ( (\*nodo)->valor != valor) {  eliminarEnListaPorValor(&(\*nodo)->sig, valor);  }  else {*//ENTONCES ENCONTRE EL VALOR*  aux = (\*nodo);  \*nodo = (\*nodo)->sig;  free(aux);  }  } }  void imprimirListaI(tNodo lista) {  *// Imprimir Iterativo*  while(lista!=NULL) {  printf("\n valor=%4d, dirNodo:<%p>, dirSig:<%p>",(lista)->valor,lista,lista->sig);  lista=lista->sig;  } } void imprimirListaR(tNodo lista) {  *// Imprimir Recursivo*  if(lista!=NULL){  printf("\n valor=%4d, dirNodo:<%p>, dirSig:<%p>",(lista)->valor,lista,lista->sig);  imprimirListaR(lista->sig);  } } void imprimirListaInversaR(tNodo lista) {  if(lista!=NULL) {  imprimirListaInversaR(lista->sig);  printf("\n valor=%4d, dirNodo:<%p>, dirSig:<%p>",(lista)->valor,lista,lista->sig);  } }**  **int largoLista(tNodo lista){   *// Iterativo*  int res=0;  while(lista!=NULL) {  res++;  lista=lista->sig;  }   return res; }  void eliminarLista(tNodo\* nodo){  *// elimina todos los elementos de una lista*  *// Y por último lista queda en NULL*  *// recursiva*  if ( \*nodo != NULL) {  eliminarLista(&(\*nodo )->sig);  tNodo aux = \*nodo;  \*nodo = aux->sig;  free(aux);  }  }   int esListaVacia(tNodo lista){  return lista==NULL; }   int eliminarEnListaPorPosicion(tNodo\* nodo, int pos){  int valor=0;  tNodo aux = NULL;  if ( \*nodo != NULL ) {  if ( pos > 0) {  valor = eliminarEnListaPorPosicion(&(\*nodo)->sig, pos-1);  }  else if( pos == 0) {*//ENTONCES ENCONTRÉ LA POSICIÓN*   aux = (\*nodo);  valor = aux->valor;  \*nodo = (\*nodo)->sig;  free(aux);  }  }  return valor;  }** |
| --- |

**SALIDA**

| L I S T A S  ================================================== Agrega valores a lista [10,20,30] --------------------------------------------------   valor= 10, dirNodo:<00711B70>, dirSig:<00711B80>  valor= 20, dirNodo:<00711B80>, dirSig:<00711B90>  valor= 30, dirNodo:<00711B90>, dirSig:<00000000>  ================================================== inserta adelante el 114 y luego el 7 resultando la lista [7,114,10,20,30] --------------------------------------------------   valor= 7, dirNodo:<00713F78>, dirSig:<00711BA0>  valor= 114, dirNodo:<00711BA0>, dirSig:<00711B70>  valor= 10, dirNodo:<00711B70>, dirSig:<00711B80>  valor= 20, dirNodo:<00711B80>, dirSig:<00711B90>  valor= 30, dirNodo:<00711B90>, dirSig:<00000000>  ================================================== De la lista anterior se elimina el valor 114 resultando la lista [7,10,20,30] --------------------------------------------------   valor= 7, dirNodo:<00713F78>, dirSig:<00711B70>  valor= 10, dirNodo:<00711B70>, dirSig:<00711B80>  valor= 20, dirNodo:<00711B80>, dirSig:<00711B90>  valor= 30, dirNodo:<00711B90>, dirSig:<00000000>  ================================================== inserta ordenado el 22 y el 3 en la lista resultando la lista [3,7,10,20,22,30] --------------------------------------------------   valor= 3, dirNodo:<00713F88>, dirSig:<00713F78>  valor= 7, dirNodo:<00713F78>, dirSig:<00711B70>  valor= 10, dirNodo:<00711B70>, dirSig:<00711B80>  valor= 20, dirNodo:<00711B80>, dirSig:<00711BA0>  valor= 22, dirNodo:<00711BA0>, dirSig:<00711B90>  valor= 30, dirNodo:<00711B90>, dirSig:<00000000>  ================================================== De la lista anterior Se elimina la posición 1 -------------------------------------------------- -- Se elimino la posicion 1 y el valor -- extraido es7 -- y se vuelve imprimir   valor= 3, dirNodo:<00713F88>, dirSig:<00711B70>  valor= 10, dirNodo:<00711B70>, dirSig:<00711B80>  valor= 20, dirNodo:<00711B80>, dirSig:<00711BA0>  valor= 22, dirNodo:<00711BA0>, dirSig:<00711B90>  valor= 30, dirNodo:<00711B90>, dirSig:<00000000>  ================================================== De la lista anterior Se inserta en posicion 1 el valor 111 --------------------------------------------------  -- insertando ---   valor= 3, dirNodo:<00713F88>, dirSig:<00713F78>  valor= 111, dirNodo:<00713F78>, dirSig:<00711B70>  valor= 10, dirNodo:<00711B70>, dirSig:<00711B80>  valor= 20, dirNodo:<00711B80>, dirSig:<00711BA0>  valor= 22, dirNodo:<00711BA0>, dirSig:<00711B90>  valor= 30, dirNodo:<00711B90>, dirSig:<00000000>  ================================================== La lista anterior la imprime invertida --------------------------------------------------   valor= 30, dirNodo:<00711B90>, dirSig:<00000000>  valor= 22, dirNodo:<00711BA0>, dirSig:<00711B90>  valor= 20, dirNodo:<00711B80>, dirSig:<00711BA0>  valor= 10, dirNodo:<00711B70>, dirSig:<00711B80>  valor= 111, dirNodo:<00713F78>, dirSig:<00711B70>  valor= 3, dirNodo:<00713F88>, dirSig:<00713F78>  ==================================================  Cuenta los elemento de la lista 'como el len'  Elimina la lista, ojo elimina los elementos  Verifica si la lista está vacia -------------------------------------------------- Largo de la lista = 6 ... Eliminando lista ...  :) La lista esta VACIA Largo de la lista = 0  << El programa ha finalizado: codigo de salida: 0 >> << Presione enter para cerrar esta ventana >> |
| --- |

## ELIMINAR todas la ocurrencias de un valor en una lista

| **void** **eliminarEnListaPorValorT**(tNodo\* nodo, **int** valor) {  tNodo aux = NULL;  **if** ( \*nodo != NULL) {  **if** ( (\*nodo)->valor != valor) {  eliminarEnListaPorValorT(&(\*nodo)->sig, valor);  }  **else** {*//ENTONCES ENCONTRE EL VALOR*  aux = (\*nodo);  \*nodo = (\*nodo)->sig;  free(aux);  eliminarEnListaPorValorT(nodo, valor);  }  } } |
| --- |