```
#include<iostream.h>
#include<conio.h>
struct nodo{
            int nro;
            struct nodo *sgte;
           };
void main()
{ struct nodo NodoCabecera;
  struct nodo *q;
  int i, n;
  NodoCabecera.sgte=NULL;
  cout<<"Numero de elementos:";
  cin>>n;
  for(i=0;i<n;i++)
  {cout<<"Elemento:"<<(i+1)<<endl;
   q=new(struct nodo);
   cin>>q->nro;
   q->sgte=NodoCabecera.sgte;
   NodoCabecera.sgte=q;
  }
  cout<<endl<<"Listado:"<<endl;
  q=NodoCabecera.sgte;
  while(q!=NULL)
  {cout<<q->nro<<endl;
  q=q->sgte;
  }
  getch();
}
```

Imagen 1.Ingresa y muestra lista.

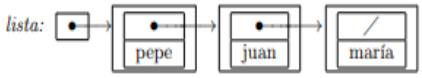
```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#define lim 100
struct nodo{
             int nro;
             struct nodo*sgte;
             };
typedef struct nodo*Tlista;
void insertar(Tlista &, int);
void unir(Tlista &, Tlista&, Tlista&);
void imprimir(Tlista);
void main()
{
 Tlista lista1=NULL;
 Tlista lista2=NULL;
 Tlista lista3=NULL;
 int opc,valor1,valor2;
 do
 {
  cout<<"1. Insertar "<<endl;
  cout<<"2. Unir"<<endl;
  cout<<"3. ver lista"<<endl;
  cout<<"4. Salir"<<endl;
  cout<<"Ingrese la opcion:";
  cin>>opc;
  switch(opc)
 {
  case 1 :
          cout<<"cantidad de elementos para la lista A:"<<endl;
          cin>>valor1;
          cout<<"Ingrese elementos a la lista:"<<endl;
          insertar(lista1, valor1);
          cout<<"cantidad de Elementos para la Lista B:"<<endl;
          cin>>valor2;
          cout<<"Ingrese elementos a la lista:"<<endl;
          insertar(lista2,valor2);
          break;
  case 2 :
          unir(lista1,lista2,lista3);
          cout<<endl<<"UNION DE LA LISTA A Y B"<<endl;
          imprimir(lista3);
          break;
  case 3: cout<<endl<<"LISTA A"<<endl;
         imprimir(lista1);
         cout<<endl<<"LISTA B"<<endl;
         imprimir(lista2);
         break;
```

```
void imprimir(Tlista lista1)
 {
 while(lista1!=NULL)
   {cout<<lista1->nro<<endl;
    lista1=lista1->sgte;
   }
   cout<<endl;
 }

 void insertar(Tlista &lista1,int valor1)
 {Tlista t,q;
  int num,i;
  for(i=0;i<valor1;i++)
  {cin>>num;
   q=new(struct nodo);
   q->nro=num;
   q->sgte=NULL;
   if(lista1==NULL)
   lista1=q;
   else
   {
   t=lista1;
   while(t->sgte!=NULL)
   t=t->sgte;
   t->sgte=q;
   }
  }
 }
```

```
void unir(Tlista &lista1, Tlista &lista2, Tlista &lista3)
{
Tlista aux,seg,pri,afg;
aux=lista1;
seg=lista2;
pri=new(struct nodo);
pri->nro=aux->nro;
pri->sgte=NULL;
lista3=pri;
aux=aux->sgte;
while(aux!=NULL)
{pri=new(struct nodo);
pri->nro=aux->nro;
pri->sgte=NULL;
afg=lista3;

while(afg->sgte!=NULL)
 afg=afg->sgte;
afg->sgte=pri;
aux=aux->sgte;
}
while(seg!=NULL)
{pri=new(struct nodo);
pri->nro=seg->nro;
pri->sgte=NULL;
afg=lista3;

while(afg->sgte!=NULL)
afg=afg->sgte;
afg->sgte=pri;
seg=seg->sgte;
}
}
```

Imagen 2. Insertar y unir listas.

```
struct Nodo ;          // Declaración adelantada del tipo incompleto Nodo
typedef Nodo* PNodo ; // Definición de tipo Puntero a tipo incompleto Nodo
struct Nodo {          // Definición del tipo Nodo
    PNodo sig ;        // Enlace a la siguiente estructura dinámica
    string dato ;      // Dato almacenado en la lista
} ;
void escribir(PNodo lista)
{
    PNodo ptr = lista;
    while (ptr != NULL) {
        cout << ptr->dato << endl ;
        ptr = ptr->sig ;
    }
}
PNodo buscar(PNodo lista, const string& dt)
{
    PNodo ptr = lista ;
    while ((ptr != NULL)&&(ptr->dato != dt)) {
        ptr = ptr->sig ;
    }
    return ptr ;
}
PNodo leer_inversa()
{
    PNodo lista = NULL ;
    string dt ;
    cin >> dt ;
    while (dt != "fin") {
        PNodo ptr = new Nodo ;
        ptr->dato = dt ;
        ptr->sig = lista ;
        lista = ptr ;
        cin >> dt ;
    }
    return lista ;
}
```

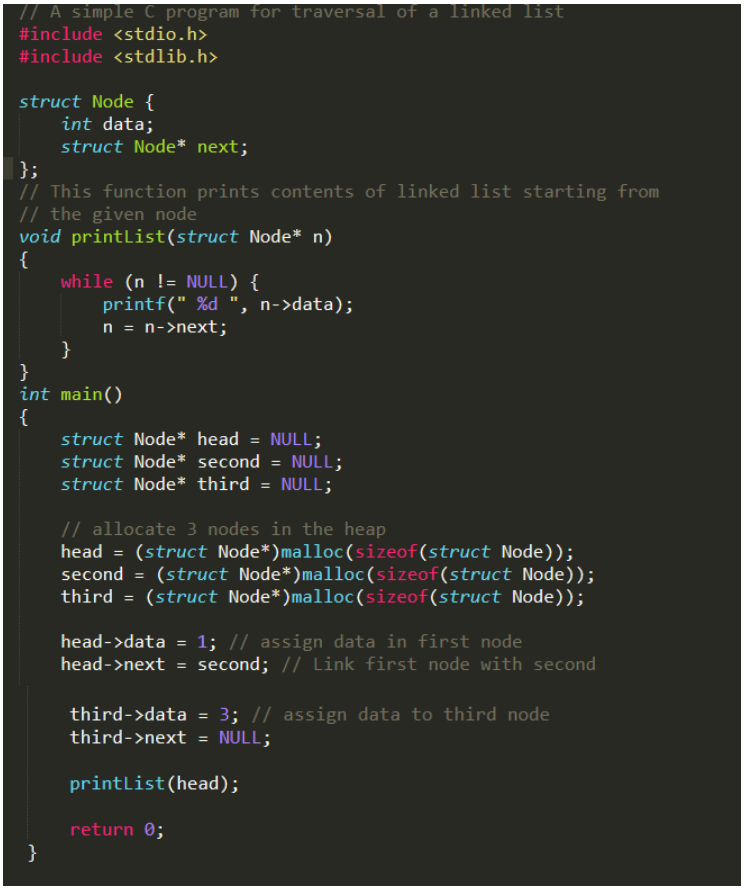



```
 void destruir(PNodo& lista)
 {
     while (lista != NULL) {
         PNodo ptr = lista ;
         lista = lista->sig ;
         delete ptr ;
     }
 }
 int main()
 {
     PNodo lista ;
     lista = leer_inversa() ;
     escribir(lista) ;
     PNodo ptr = buscar(lista, "juan");
     if (ptr != NULL) {
         cout << ptr->dato << endl;
     }
     destruir(lista) ;
 }
```

Imagen 3. Listas enlazadas lineales.

```
#include <cstddef>
#include <iostream>

using namespace std;

class Node {
    public:
        int data;
    Node * next;
};

void print_list(Node * n) {
    cout << "\nPrinting new list ... " << endl;
    while (n != NULL) {
        cout << n->data << " ";
        n = n->next;
    }
}
void push(struct Node ** head_ref, int new_data) {
    struct Node * new_node = (struct Node * ) malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = ( * head_ref);
    ( * head_ref) = new_node;
}

int main() {
    Node * head = NULL;
    Node * second = NULL;
    Node * third = NULL;

    head = new Node();
    second = new Node();
    third = new Node();

    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = NULL;

    print_list(head);
    push(&head, 11);
    print_list(head);
}
```

Imagen 4. Insertar al inicio.



```c
// A simple C program for traversal of a linked list
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};
// This function prints contents of linked list starting from
// the given node
void printList(struct Node* n)
{
    while (n != NULL) {
        printf(" %d ", n->data);
        n = n->next;
    }
}
int main()
{
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;

    // allocate 3 nodes in the heap
    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

    head->data = 1; // assign data in first node
    head->next = second; // Link first node with second

     third->data = 3; // assign data to third node
     third->next = NULL;

     printList(head);

     return 0;
 }
```

Imagen 5. A simple C program for traversal of a linked list.

```c
// A complete working C program to demonstrate all insertion methods
// on Linked List
#include <stdio.h>
#include <stdlib.h>

// A linked list node
struct Node
{
int data;
struct Node *next;
};

/* Given a reference (pointer to pointer) to the head of a list and
an int, inserts a new node on the front of the list. */
void push(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. Make next of new node as head */
    new_node->next = (*head_ref);

    /* 4. move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Given a node prev_node, insert a new node after the given
prev_node */
void insertAfter(struct Node* prev_node, int new_data)
{
    /*1. check if the given prev_node is NULL */
    if (prev_node == NULL)
    {
    printf("the given previous node cannot be NULL");
    return;
    }

    /* 2. allocate new node */
    struct Node* new_node =(struct Node*) malloc(sizeof(struct Node));

    /* 3. put in the data */
    new_node->data = new_data;

    /* 4. Make next of new node as next of prev_node */
    new_node->next = prev_node->next;

    /* 5. move the next of prev_node as new_node */
    prev_node->next = new_node;
}
```

```c
/* Given a reference (pointer to pointer) to the head
of a list and an int, appends a new node at the end */
void append(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));

    struct Node *last = *head_ref; /* used in step 5*/

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. This new node is going to be the last node, so make next of
        it as NULL*/
    new_node->next = NULL;

    /* 4. If the Linked List is empty, then make the new node as head */
    if (*head_ref == NULL)
    {
    *head_ref = new_node;
    return;
    }

    /* 5. Else traverse till the last node */
    while (last->next != NULL)
        last = last->next;

    /* 6. Change the next of last node */
    last->next = new_node;
    return;
}
// This function prints contents of linked list starting from head
void printList(struct Node *node)
{
while (node != NULL)
{
    printf(" %d ", node->data);
    node = node->next;
}
}

/* Driver program to test above functions*/
int main()
{
/* Start with the empty list */
struct Node* head = NULL;

// Insert 6. So linked list becomes 6->NULL
append(&head, 6);

// Insert 7 at the beginning. So linked list becomes 7->6->NULL
push(&head, 7);

// Insert 1 at the beginning. So linked list becomes 1->7->6->NULL
push(&head, 1);

// Insert 4 at the end. So linked list becomes 1->7->6->4->NULL
append(&head, 4);

// Insert 8, after 7. So linked list becomes 1->7->8->6->4->NULL
insertAfter(head->next, 8);

printf("\n Created Linked list is: ");
printList(head);

return 0;
}
```

Imagen 6.Display of all insertion methods

```c
// C program for generic linked list
#include<stdio.h>
#include<stdlib.h>

/* A linked list node */
struct Node
{
    // Any data type can be stored in this node
    void *data;

    struct Node *next;
};

/* Function to add a node at the beginning of Linked List.
This function expects a pointer to the data to be added
and size of the data type */
void push(struct Node** head_ref, void *new_data, size_t data_size)
{
    // Allocate memory for node
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

    new_node->data = malloc(data_size);
    new_node->next = (*head_ref);

    // Copy contents of new_data to newly allocated memory.
    // Assumption: char takes 1 byte.
    int i;
    for (i=0; i<data_size; i++)
        *(char *)(new_node->data + i) = *(char *)(new_data + i);

    // Change head pointer as new node is added at the beginning
    (*head_ref) = new_node;
}
/* Function to print nodes in a given linked list. fpitr is used
to access the function to be used for printing current node data.
Note that different data types need different specifier in printf() */
void printList(struct Node *node, void (*fptr)(void *))
{
    while (node != NULL)
    {
        (*fptr)(node->data);
        node = node->next;
    }
}

// Function to print an integer
void printInt(void *n)
{
printf(" %d", *(int *)n);
}

// Function to print a float
void printFloat(void *f)
{
printf(" %f", *(float *)f);
}

/* Driver program to test above function */
int main()
{
    struct Node *start = NULL;

    // Create and print an int linked list
    unsigned int_size = sizeof(int);
    int arr[] = {10, 20, 30, 40, 50}, i;
    for (i=4; i>=0; i--)
    push(&start, &arr[i], int_size);
    printf("Created integer linked list is \n");
    printList(start, printInt);

    // Create and print a float linked list
    unsigned float_size = sizeof(float);
    start = NULL;
    float arr2[] = {10.1, 20.2, 30.3, 40.4, 50.5};
    for (i=4; i>=0; i--)
    push(&start, &arr2[i], float_size);
    printf("\n\nCreated float linked list is \n");
    printList(start, printFloat);

    return 0;
}
```

Imagen 7.Generic linked list

Bibliografía:

[Imagen 1]: *Rec Pro Digital- Tecnología Programación Digital*. [Disponible en línea]: https://tecpro-digital.com/programacion-en-c-listas-enlazadas-ingresa-y-muestra/ [Acceso: 29 Agosto 2020].

[Imagen 2]: *Rec Pro Digital- Tecnología Programación Digital*. [Disponible en línea]: https://tecpro-digital.com/programacion-en-c-listas-enlazadas-insertar-imprimir-y-unir-listas/ [Acceso: 29 Agosto 2020].

[Imagen 3]: Vicente Benjumea, Manuel Roldán,"Fundamentos de Programación con Lenguaje de Programación C++", Universidad de Málaga, Dpto. Lenguajes y CC. Computación E.T.S.I Informática, 23 de Octubre 2017, [Disponible en línea]: http://www.lcc.uma.es/~vicente/docencia/cppdoc/programacion_cxx.pdf [Acceso: 29 Agosto 2020].

[Imagen 4]: *BitDegre learn*.- How to Use Linked List in C++ [Disponible en línea]: https://www.bitdegree.org/learn/linked-list-c-plus-plus [Acceso: 29 Agosto 2020].

[Imagen 5]: *GeeksforGeeks-Linked List* [Disponible en línea]: https://www.geeksforgeeks.org/linked-list-set-1-introduction/ [Acceso: 29 Agosto 2020].

[Imagen 6]: *GeeksforGeeks-Linked List|set 2(Inserting a node)* [Disponible en línea]: https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/?ref=lbp [Acceso: 29 Agosto 2020].

[Imagen 7]: *GeeksforGeeks-Linked-Generic Linked List in C* [Disponible en línea]: https://www.geeksforgeeks.org/generic-linked-list-in-c-2/ [Acceso: 29 Agosto 2020].