Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Capítulo IV Estructuras de Datos Lineales Dinámicas.

Contenido.

Introducción

En este capítulo se comienza el estudio de las estructuras de datos dinámicas. Aprenderá a trabajar con punteros y asignación y liberación de memoria. Hasta ahora ha trabajado con estructuras de datos estáticas; donde se reserva un espacio de memoria fijo, dentro del que se debe trabajar como son los arreglos, a partir de ahora vamos a trabajar con ED dinámicas, que a continuación se nombran.

Las diferentes Listas Enlazadas que se estudian son:

- Listas Simplemente Enlazadas.
- Listas Doblemente enlazadas.
- Listas Simplemente Enlazadas circulares.
- Listas Doblemente Enlazadas Circulares.
- Listas con Nodo Cabeza.

En este semestre estudiaremos este tema, utilizando el lenguaje C++.
Por lo que a continuación se muestran los ejemplos de forma general, pero después se programan con C.

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Listas Dinámicas.

Las Listas dinámicas son un conjunto de elementos que crecen y decrecen dinámicamente en función de si se necesitan más elementos ó si no se necesitan. El trabajo con ellas es mejor comparado con los arreglos. La ventaja más notoria sobre ellos, es que la inserción y eliminación es más rápida y sólo se utiliza el espacio en memoria realmente necesario para la aplicación.

Listas Simplemente Enlazadas.

Es una lista que crece y decrece dinámicamente, cada uno de los elementos que la componen tiene un enlace al siguiente de la lista, es decir, al que le sigue.

Representación gráfica de la lista simplemente enlazada.

Gráficamente se puede representar una lista simplemente enlazada como el gráfico de la figura 1. En él se ven un conjunto de elementos, para este caso, caracteres; donde cada elemento tiene la posición de donde se encuentra el siguiente elemento.

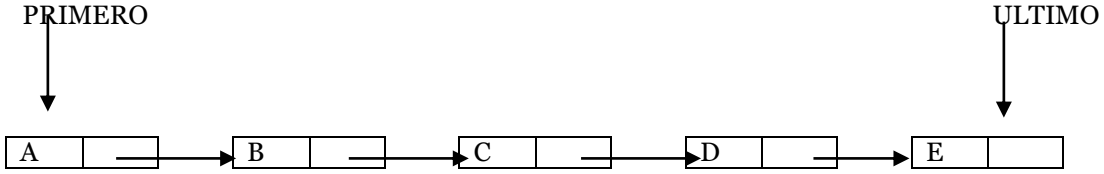

Fig1

Donde usar una Lista Simplemente Enlazada.

En cualquier problema donde se maneje un conjunto de elementos, y cada uno se encuentre a continuación de otro.

Por ejemplo: Si se tiene una lista de varias asignaturas, y de cada una de ellas se conoce el nombre, y la cantidad de horas. O una lista de estudiantes, o una lista de médicos, etc.

Representación en memoria de una Lista Simplemente Enlazada:

Como se puede apreciar de la figura anterior, la lista está formada por varios elementos, uno de ellos trabajar con las listas, ó ese conjunto de elementos, se debe poder crear la lista e ir insertando elementos. Entre las operaciones a realizar están:

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Operaciones Básicas:

Para trabajar con las listas, ó ese conjunto de elementos, se debe poder crear la lista e ir insertando elementos. Entre las operaciones a realizar están:

- **Vacía**

  Esta operación verifica que si la lista no tiene ningún elemento, en cuyo caso devuelve verdadero.

```
int vacia()
{
        if (primero==NULL)
                return 1;
        else
                return 0;

}
```

- **Insertar.**
  - Que podría hacerse al inicio de la lista, o al final de ella, o en una determinada posición.
  - A continuación, se expone el código asociado a estas operaciones de inserción

Operación que inserta un elemento por delante de la lista: Al insertar de esta forma, cuando llegue un elemento a la lista, este pasará a ser el primero de ella

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
void insertar_delante (char *valor)
{
   nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else

        {
                p->siguiente=primero;
                primero=p;
        }

return;

}
```

Operación que inserta un elemento por detrás de la lista: Al insertar de esta forma, cuando llegue un elemento a la lista, este pasará a ser el último de ella

```
void insertar_detras (char *valor)
{
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else

        {
                ultimo->siguiente=p;
                ultimo=p;
        }
        return;
}
```

Operación que inserta un elemento ordenadamente en la lista: En este caso se está insertando de menor a mayor, en orden ascendente**.** Al insertar de esta forma, cuando llegue un elemento a la lista, se buscará el lugar donde debe ir el elemento a insertar, se toma en cuenta también, que si es menor a todos deberá ser entonces el primero de la lista, si es mayor a todos, deberá ocupar el último lugar en la lista.

```
int ins_ordenadamente (char *valor)
{
        nodocaracter  *p,*q,*s;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
p->datos = valor;
p->siguiente=NULL;
if (vacia())

{
        primero=p;
        ultimo=p;
}
else
{
        q=primero;
        s=primero;
        while ((q!= NULL) && (q->datos<valor))
        {
                s=q;
                q = q->siguiente;

        }

        if (q==primero)
        {
                p->siguiente=primero;
                primero=p;
        }
        else
        {

                s->siguiente=p;
                p->siguiente=q;
                if (q==NULL)
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                                        ultimo=p;
                }


        }

        return 0;

}
```

Eliminar.

- Que también podría tener otras variantes, tales como eliminar el primero de la lista, o eliminar el último, o eliminar un elemento determinado. Vamos a exponer la operación de eliminar un determinado elemento en la lista.

```
void eliminar(char *valor)
{
        nodocaracter  *p,*q;

   if (vacia())

   {
         cout<<"La lista esta vacia";

   }
   else
   {
        q=primero;
   p=primero;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
while ((p!= NULL) && (p->datos!=valor))
{
        q=p;
        p = p->siguiente;

   }
if (p->datos==valor)
   {
        q->siguiente=p->siguiente;
        if (p==primero)
                primero=p->siguiente;
        if (p==ultimo)
                ultimo=q;

   }
   else
   cout<<"No existe ese elemento en la lista";
}

return ;

}
```

- **Mostrar.**

  - En esta operación se mostrarán los diferentes elementos que componen la lista. Si la lista esta vacía, se muestra un mensaje de error, caso contrario se recorre y se va mostrando uno a uno cada elemento de ella, hasta llegar al final de la lista.

```
void mostrar ()
{
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        nodocaracter  *p;
        if (vacia())
                cout<<"Lista vacia";
        else

        {
                p = primero     ;
                while (p != NULL){
                        cout<< p->datos;
                        p = p->siguiente;
                      }
       }

                return;
}
```

Otras operaciones que se podrían implementar serian:

- Ir al Primero.
- Ir al Último.
- Ir al Anterior.
- Ir al siguiente.
- Buscar un determinado elemento.
- Ordenar la lista.
- Máximo.
- Minimo
- Etc.

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Implementación de una Lista Simplemente Enlazada utilizando C.

A continuación, se muestra el código de una lista simplemente enlazada de caracteres:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>

using namespace std;


typedef struct nodo {
        char   *datos;
        struct nodo *siguiente;
}nodocaracter;

nodocaracter *primero,*ultimo;

void iniciar()
{
primero=NULL;
ultimo=NULL;
}

int vacia()
{
        if (primero==NULL)
                return 1;
        else
                return 0;

}

void insertar_delante (char *valor)
{
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
   nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else

        {
                p->siguiente=primero;
                primero=p;
        }

return;

}




void insertar_detras (char *valor)
{

        nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        {
                ultimo->siguiente=p;
                ultimo=p;
        }
        return;
}
```

```
void mostrar ()
{

        nodocaracter  *p;
        if (vacia())
                cout<<"Lista vacia";
        else

        {
                p = primero      ;
                while (p != NULL){
                        cout<< p->datos;
                        p = p->siguiente;
                    }
  }

                return;
}
```

```
int ins_ordenadamente (char *valor)
{
        nodocaracter  *p,*q,*s;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
p->siguiente=NULL;
if (vacia())

{
        primero=p;
        ultimo=p;
}
else
{
        q=primero;
        s=primero;
        while ((q!= NULL) && (q->datos<valor))
        {
                s=q;
                q = q->siguiente;

        }

        if (q==primero)
        {
                p->siguiente=primero;
                primero=p;
        }
        else
        {

                s->siguiente=p;
                p->siguiente=q;
                if (q==NULL)
                                ultimo=p;
        }


}

return 0;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
}



void eliminar(char *valor)
{
        nodocaracter  *p,*q;

  if (vacia())

  {
         cout<<"La lista esta vacia";

  }
  else
  {
        q=primero;
  p=primero;
        while ((p!= NULL) && (p->datos!=valor))
        {
                q=p;
                p = p->siguiente;

          }
        if (p->datos==valor)
          {
                q->siguiente=p->siguiente;
                if (p==primero)
                        primero=p->siguiente;
                if (p==ultimo)
                        ultimo=q;

          }
          else
          cout<<"No existe ese elemento en la lista";
        }
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
return ;

}

int main(int argc, char *argv[])
        {
        iniciar();
        insertar_delante("A");
        insertar_delante("B");
        insertar_delante("C");
        mostrar();
        eliminar("B");
        mostrar();
        return 0;
}
```

Listas SE Circulares.

Que son

Una lista SE circular es una lista Simplemente enlazada, pero que además es circular, en la que el último elemento enlaza con el primero. Es posible acceder a cualquier elemento de la lista circular desde cualquier punto dado. Las operaciones en listas circulares son similares a las estudiadas anteriormente, con la diferencia que los nodos apuntan siempre a un nodo, nunca a NULL.

En estas listas no existe ni primero ni último elemento, aunque se debe elegir obligatoriamente un puntero para referenciar la lista.

Tiene el inconveniente de que requiere un diseño cuidadoso para evitar caer en un bucle Infinito.

Podemos ver ya sea listas simplemente enlazadas circulares o listas doblemente enlazadas circulares.

Representación gráfica de la Lista Simplemente Enlazada Circular.

Gráficamente se puede representar una lista simplemente enlazada como el gráfico de la figura 2. En él se ven un conjunto de elementos, para este caso, caracteres; donde cada elemento tiene la posición de donde se encuentra el siguiente elemento.

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

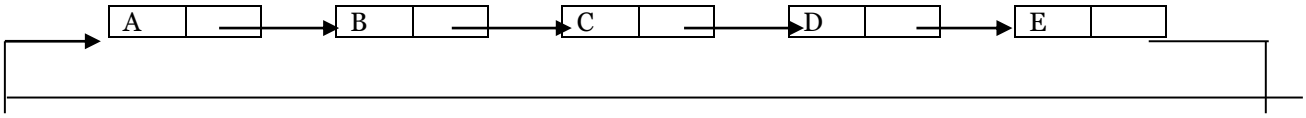

Fig2

Operaciones Básicas de la Lista SE Circular

El trabajo con estas listas, es similar al trabajo con las anteriores, pero tomando en cuenta que ahora es circular, o sea, el último elemento de la lista apuntará al primero.
Quedando el código de la misma de la siguiente forma:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>

using namespace std;


typedef struct nodo {
        char   *datos;
        struct nodo *siguiente;
}nodocaracter;

nodocaracter *primero,*ultimo;

void iniciar()
{
        primero=NULL;
        ultimo=NULL;
}

int vacia()
{
        if (primero==NULL)
                return 1;
        else
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                return 0;

}

void insertar_delante (char *valor)
{
        nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=p;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else

        {
                p->siguiente=primero;
                primero=p;
                ultimo->siguiente=p;
        }

        return;

}




void insertar_detras (char *valor)
{

        nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=p;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else

        {
                ultimo->siguiente=p;
                ultimo=p;
                ultimo->siguiente=primero;
        }
        return;
}


void mostrar ()
{

        nodocaracter  *p;
        if (vacia())
                cout<<"Lista vacia";
        else

        {
                p = primero      ;
                do
                {
                        cout<< p->datos;
                        p = p->siguiente;
                }
                while (p!=primero);
        }

        return;
}
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
int insertar_ordenadamente (char *valor)
{
        nodocaracter  *p,*aux1,*aux2;
        int terminar;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=p;
        if (vacia())

        {
                primero=p;
                ultimo=p;
        }
        else
        {
                aux1=primero;
                aux2=primero;
                terminar=0;
                do
                {
                        if (aux1->datos<valor)
                                terminar=1;
                        else
                          {
                                aux2=aux1;
                          aux1=aux1->siguiente;
                          }
                }
                while((aux1!= primero) && (!terminar));

                if (aux1==aux2)
                  {
                        p->siguiente=aux1;
                  ultimo->siguiente=p;
                        primero=p;
                  }
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                else
                  {
                                aux2->siguiente=p;
                                p->siguiente=aux1;
                                if (aux1==primero);
                                ultimo=p;
                        }

        }

        return 0;

}

void eliminar(char *valor)
{
        nodocaracter  *p,*aux;
        int terminar;
        if (vacia())

        {
                cout<<"La lista esta vacia";

        }
        else
        {
                aux=primero;
                p=primero;
                terminar=0;
                do
                {
                        if (p->datos==valor)
                                terminar=1;
                        else
                        {
                                aux=p;
                                p=p->siguiente;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                    }
                }
                while((p!= primero) && (!terminar));

                if (p->datos==valor)
                  {

                    if (p==primero)
                        {
                            primero=primero->siguiente;
                                    ultimo->siguiente=primero;
                        }
                    else if (p==ultimo)
                        {
                              aux->siguiente=p->siguiente;
                        ultimo=aux;
                        }
                    else
                            aux->siguiente=p->siguiente;
                  }
                else
                    cout<<"No existe ese elemento en la lista";
        }

        return ;

}



int main(int argc, char *argv[])
{
        iniciar();
        insertar_delante("A");
        insertar_delante("B");
        insertar_delante("C");
        mostrar();
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        eliminar("C");
        mostrar();
return 0;
}
```

Listas Doblemente Enlazadas.

Es una lista que crece y decrece dinámicamente, cada uno de los elementos que la componen tiene un enlace al siguiente, es decir al que le sucede y otro enlace, al que le antecede.

Una lista doblemente enlazada es muy parecida a la lista estudiada en el apartado anterior. El tratamiento es el mismo con la diferencia de que en este caso, cada nodo va a tener dos enlaces, uno al antecesor y otro al sucesor. Esta lista tiene asociado tres campos donde el primero apunta al antecesor, el campo de información que contiene la información del nodo, y el tercer campo apuntará al siguiente nodo de la lista.

En resumen los campos son los siguientes:

- Enlace izquierdo
- Información
- Enlace derecho

Representación gráfica de la lista doblemente enlazada

Gráficamente se puede representar una lista doblemente enlazada como el gráfico de la figura 3. En él se ven un conjunto de elementos, donde cada elemento tiene la posición de donde se encuentra el siguiente elemento y donde se encuentra el anterior.

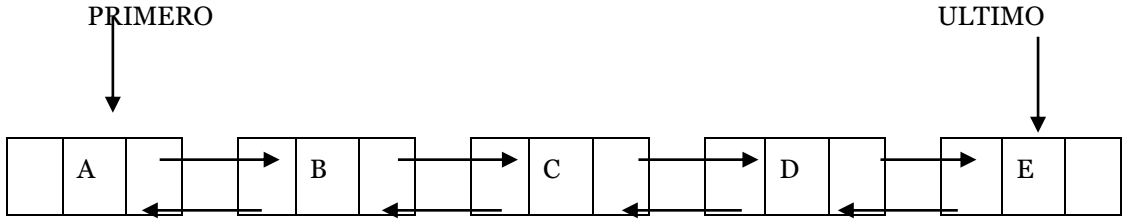

Fig. 3

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Operaciones Básicas de la lista DE

El trabajo con estas listas, es similar al trabajo con las anteriores, pero tomando en cuenta que ahora tienen un enlace más.

Las operaciones básicas serían lo mismo también.

Las operaciones Básicas serán las mismas estudiadas con anterioridad, con la diferencia de que su implementación no es la misma.

Quedando entonces el código de la siguiente forma:

Definición en memoria de la lista doblemente enlazada

Se define un nodo, con un campo información llamado datos de tipo carácter, y dos campos de tipo puntero, uno enlazará al nodo que le antecede y otro enlazará al nodo que le sucede.

```
typedef struct nodo {
        char   *datos;
        struct nodo *siguiente, *anterior;
}nodocaracter;

nodocaracter *primero,*ultimo;
```

Implementación de las Operaciones Básicas de la LDE

Operación para Inicializar la Lista.

```
void iniciar()
{
        primero=NULL;
        ultimo=NULL;
}
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Operación vacía.

La lista estará vacía cuando no tenga elementos, no existe un primero en ella

```
int vacia()
{
        if (primero==NULL)
                return 1;
        else
                return 0;

}
```

Operación Insertar.

- La operación se implementa en un procedimiento que recibe como parámetro el valor que se quiere insertar.
- Es igual que el insertar de la lista simplemente enlazada, pero en este caso habría que actualizar el enlace anterior del nodo.
- Se implementan diferentes formas de inserción.
- Quedando de la siguiente forma:

```
void insertar_delante (char *valor)
{
        nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        p->anterior=NULL;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        {
                p->siguiente=primero;
                primero->anterior=p;
                primero=p;
        }

        return;

}



void insertar_detras (char *valor)
{

        nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        p->anterior=NULL;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else

        {
                ultimo->siguiente=p;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                p->anterior=ultimo;
                ultimo=p;
        }
        return;
}

int ins_ordenadamente (char *valor)
{
        nodocaracter  *p,*q,*s;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        p->anterior=NULL;
        if (vacia())

        {
                primero=p;
                ultimo=p;
        }
        else
        {
                q=primero;
                s=primero;
                while ((q!= NULL) && (q->datos<valor))
                {
                        s=q;
                        q = q->siguiente;

                }

                if (q==s)
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                {
                        p->siguiente=primero;
                        primero->anterior=p;
                        primero=p;
                }
               else
                {

                  s->siguiente=p;
                  p->anterior=s;
                  p->siguiente=q;
                  if (q!=NULL)
                           q->anterior=p;
                  else
                        ultimo=p;
                  }


        }

        return 0;

}
```

Operación de Eliminar

- Primero se verifica si la lista está vacía, en cuyo caso no hay nada que eliminar, dado que primero no apunta a nada.
- Se busca el elemento y se elimina.

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
void eliminar(char *valor)
{
        nodocaracter  *p,*q;
        if (vacia())
        {
                cout<<"La lista esta vacia";
        }
        else
        {
                q=primero;
                p=primero;
                while ((p!= NULL) && (p->datos!=valor))
                {
                        q=p;
                        p = p->siguiente;

                }
                if (p->datos==valor)
                {

                        if (p==primero)
                        {

                                primero=primero->siguiente;
                                primero->anterior=NULL;

                        }
                        else if (p==ultimo)
                                        {
                                        q->siguiente=NULL;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                                        ultimo=p;

                                    }
                                else
                                        {
                                        q->siguiente=p->siguiente;
                                        p->siguiente->anterior=q;
                                    }
                            }
                            else
                                cout<<"No existe ese elemento en la lista";
                        }

                        return ;

                }
```

Implementación de una Lista Doblemente Enlazada utilizando C.

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>

using namespace std;


typedef struct nodo {
        char   *datos;
        struct nodo *siguiente, *anterior;
}nodocaracter;

nodocaracter *primero,*ultimo;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
void iniciar()
{
        primero=NULL;
        ultimo=NULL;
}

int vacia()
{
        if (primero==NULL)
                return 1;
        else
                return 0;

}

void insertar_delante (char *valor)
{
        nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        p->anterior=NULL;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else

        {
                p->siguiente=primero;
                primero->anterior=p;
                primero=p;
        }
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        return;

}

void insertar_detras (char *valor)
{

        nodocaracter  *p;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        p->anterior=NULL;
        if (vacia())
        {
                primero=p;
                ultimo=p;
        }
        else

        {
                ultimo->siguiente=p;
                p->anterior=ultimo;
                ultimo=p;
        }
        return;
}
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
void mostrar_delante_atras ()
{

        nodocaracter  *p;
        if (vacia())
                cout<<"Lista vacia";
        else

        {
                p = primero        ;
                while (p != NULL){
                        cout<< p->datos;
                        p = p->siguiente;
                }
        }

        return;
}

void mostrar_de_atras_alante ()
{

        nodocaracter  *p;
        if (vacia())
                cout<<"Lista vacia";
        else

        {
                p = ultimo         ;
                while (p != NULL){
                        cout<< p->datos;
                        p = p->anterior;
                }
        }

        return;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
}


int ins_ordenadamente (char *valor)
{
        nodocaracter  *p,*q,*s;
        p= (nodocaracter*)(malloc (sizeof (nodocaracter)));
        p->datos = valor;
        p->siguiente=NULL;
        p->anterior=NULL;
        if (vacia())

        {
                primero=p;
                ultimo=p;
        }
        else
        {
                q=primero;
                s=primero;
                while ((q!= NULL) && (q->datos<valor))
                {
                        s=q;
                        q = q->siguiente;

                }

                if (q==s)
                  {
                        p->siguiente=primero;
                        primero->anterior=p;
                        primero=p;
                  }
                else
                  {
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                    s->siguiente=p;
                    p->anterior=s;
                    p->siguiente=q;
                    if (q!=NULL)
                            q->anterior=p;
                    else
                            ultimo=p;
                    }


        }

        return 0;

}

void eliminar(char *valor)
{
        nodocaracter  *p,*q;

        if (vacia())

        {
                cout<<"La lista esta vacia";

        }
        else
        {
                q=primero;
                p=primero;
                while ((p!= NULL) && (p->datos!=valor))
                {
                        q=p;
                        p = p->siguiente;

                }
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                if (p->datos==valor)
                {

                        if (p==primero)
                        {

                                primero=primero->siguiente;
                                primero->anterior=NULL;

                        }
                        else if (p==ultimo)
                                        {
                                        q->siguiente=NULL;
                                        ultimo=p;

                              }
                        else
                                      {
                                       q->siguiente=p->siguiente;
                                       p->siguiente->anterior=q;
                                   }
                }
                else
                        cout<<"No existe ese elemento en la lista";
        }

        return ;

}



int main(int argc, char *argv[])
{
        iniciar();
        insertar_delante("A");
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        insertar_delante("B");
        insertar_delante("C");
        mostrar_delante_atras();
        eliminar("C");
        mostrar_delante_atras();




        return 0;
}
```

Listas Doblemente Enlazadas Circulares.

Las listas doblemente enlazadas también pueden ser circulares. Las listas son estructuras muy ricas y variadas.

En muchas aplicaciones se requiere recorrer la lista en ambas direcciones. Estos recorridos se pueden conseguir manteniendo dos campos de enlace en cada nodo en lugar de un solo enlace como se vio al estudiar las listas simplemente enlazadas. Estos enlaces se utilizan para denotar la dirección del predecesor y sucesor de un nodo dado. El predecesor se llama enlace izquierdo o anterior y el sucesor enlace derecho o siguiente.

Una lista circular doblemente enlazada, es una lista en la que cada nodo apunta al siguiente y al anterior y donde el "ultimo" elemento apunta al primer elemento y el primero apunta al último nodo. En estas listas no existe ni primero ni último aunque se debe elegir obligatoriamente un puntero para referenciar la lista.

Representación gráfica de la Lista Doblemente Enlazada Circular

Gráficamente se puede representar una lista doblemente enlazada circular como el gráfico de la figura 4. En él se ven un conjunto de elementos, para este caso, caracteres; donde cada elemento tiene la posición de donde se encuentra el siguiente elemento y donde se encuentra el que le antecede; y el último tiene la dirección del primero, como el primero, tiene la dirección del último.

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

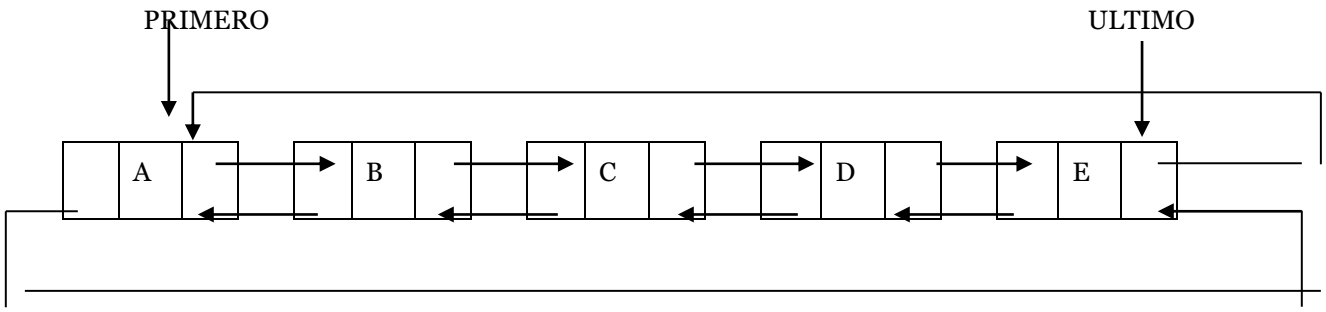

Fig. 4

Operaciones Básicas

Para trabajar con las listas, ó ese conjunto de elementos, se debe poder crear la lista e ir insertando elementos. Entre las operaciones a realizar se tienen las mismas estudiadas anteriormente, tomando en cuenta la diferencias entre ellas, modificando las operaciones, pues al insertar y/o eliminar elementos en la lista, se deben actualizar correctamente los enlaces.

Debe tomar en cuenta que es una lista Doblemente Enlazada, pero que ahora también es circular.

Listas con Nodos Cabezas.

Muchas veces es necesario trabajar con un nodo extra de la lista, que pueda dar información general de ella, como, por ejemplo, la cantidad de elementos que tiene. Este nodo se llama Nodo Cabeza. Se puede usar como ya se dijo anteriormente para dar alguna información general de la lista ó sencillamente para marcar el inicio de ella.

Podemos utilizar los nodos cabezas cada vez que sea necesario y se pueden asociar a:

- Listas Simplemente Enlazadas
- Listas Doblemente Enlazadas
- Listas Simplemente Enlazadas Circulares
- Listas Doblemente Enlazadas Circulares

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Listas Simplemente Enlazada con Nodo Cabeza.

Es una Lista simplemente enlazada que se le adiciona un nodo más, el nodo cabeza.

Representación gráfica de la lista simplemente enlazada con Nodo Cabeza

Gráficamente se puede representar una lista simplemente enlazada con nodo cabeza como el gráfico de la figura 5. En él se ven un conjunto de elementos, para este caso, caracteres; donde cada elemento tiene la posición de donde se encuentra el siguiente elemento, pero además se tiene un nodo extra, el nodo cabeza.

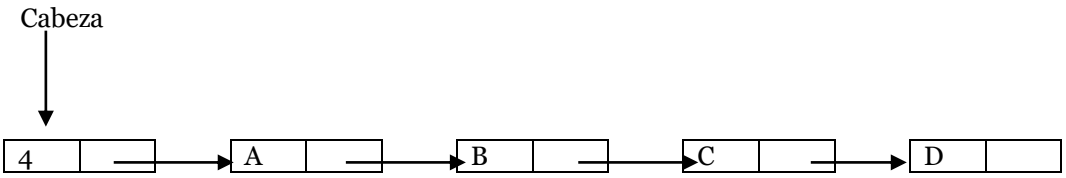

Fig. 5

Listas Doblemente Enlazada con Nodo Cabeza

Una lista doblemente enlazada con nodo cabeza es una lista doblemente enlazada a la cual se le adiciona otro nodo al inicio de la misma, es decir un nodo cabeza que va a tener información general de la lista y va a marcar el inicio de la misma.

Representación gráfica de la Lista Doblemente Enlazada con Nodo Cabeza.

Gráficamente se puede representar una lista doblemente enlazada con nodo cabeza como el gráfico de la figura 6. En él se ven un conjunto de elementos, para este caso, caracteres; donde cada elemento tiene la posición de donde se encuentra el siguiente elemento.

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

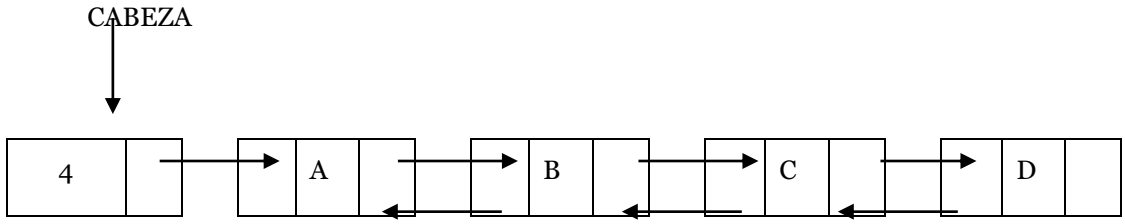

Fig. 6

Listas Simplemente Enlazada Circular con Nodo Cabeza

Una lista simplemente enlazada circular con nodo cabeza no es más que una lista simplemente enlazada circular , vista con anterioridad, pero que además se le adiciona un nodo, que va a tener una información general de la lista o servirá para tener la referencia de donde comienza esta, llamado nodo cabeza.

Representación gráfica de la lista simplemente enlazada Circular con Nodo Cabeza

Gráficamente se puede representar una lista simplemente enlazada circular con nodo cabeza como el gráfico de la figura 7. En él se ven un conjunto de elementos, para este caso, caracteres; donde cada elemento tiene la posición de donde se encuentra el siguiente elemento.

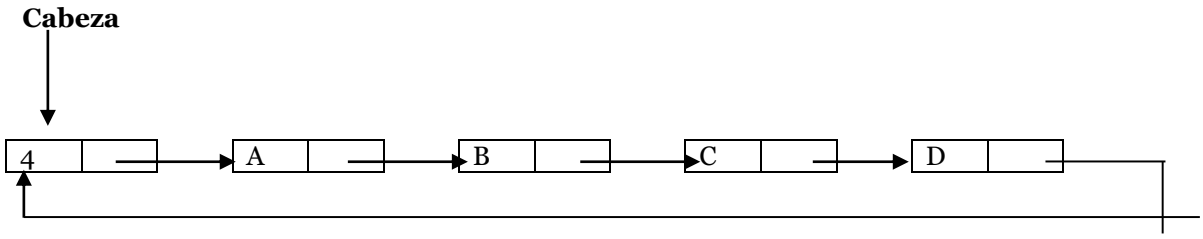

Fig. 7

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Aplicaciones de las Listas Enlazadas.

Las listas tienen muchísimas aplicaciones, pues como usted ya ha aprendido una lista es una colección de elementos, y en cualquier trabajo de la vida real que se quiera llevar a la computadora, por lo general, no va a ser un solo elemento que lo compone, sino que habrá muchos.
Por ejemplo alguna vez ha ido al hospital y habrá visto que hay un gran cantidad de médicos, también hay un sin número de especialidades en las que usted puede atenderse; es decir que si queremos representar esa situación del mundo real en la computadora , necesariamente se debe utilizar una lista.

Entre las aplicaciones de las listas enlazadas podemos mencionar las siguientes:

- Multilistas
- Resolución de operaciones matemáticas como polinomios.
- En toda actividad que se requiera almacenar o llevar el control de más de un dato.

Desarrollo de un Ejemplo Utilizando una Lista SE

- Para el ejemplo a desarrollar se ha elegido trabajar con una lista de asignaturas, donde la información de cada nodo es:
  - Código de la Asignatura.
  - Nombre de la Asignatura.
  - Cantidad de Horas.
- Para ubicarnos en el ejemplo, podemos representarnos una lista de la siguiente forma:

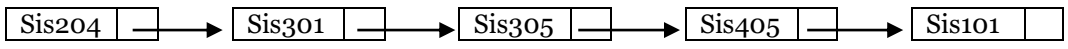

Fig8

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>


using namespace std;


struct nodoasig {
        string codigo, descripcion;
        int cantidad;
        struct nodoasig *siguiente;
};



nodoasig *primero,*ultimo;

void iniciar()
{
        primero=NULL;
        ultimo=NULL;
}

int vacia()
{
        if (primero==NULL)
                return 1;
        else
                return 0;

}

void insertar_delante ( string cod, string nomb, int cant)
{
        nodoasig *nodo;

        nodo= new nodoasig;
        nodo->siguiente=NULL;
        nodo->codigo=cod;
        nodo->descripcion=nomb;
        nodo->cantidad=cant;

        if (vacia())
        {


                primero=nodo;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
                ultimo=nodo;

        }
        else

        {
                nodo->siguiente=primero;
                primero=nodo;
        }

        return;

}




void mostrar ()
{

        nodoasig *p;

        if (vacia())
                cout<<"Lista vacia";
        else

        {
                p = primero     ;
                while (p != NULL){
                        cout<< p->codigo<<endl;
                        cout<< p->descripcion<<endl;
                        cout<< p->cantidad<<endl;
                        p = p->siguiente;
                        getchar();
                }
        }

        return;
}



void eliminar(string valor)
{
        nodoasig  *p,*q;

        q=primero;
        p=primero;
        while ((p!= NULL) && (p->codigo!=valor))
        {
                q=p;
                p = p->siguiente;
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
        }
        if (p==NULL)
                cout<<"No existe ese elemento en la lista";
        else
        {

                if (primero==ultimo)
                {
                        primero=NULL;
                        ultimo=NULL;
                }
                else
                {
                        q->siguiente=p->siguiente;
                        if (p==primero)
                                primero=p->siguiente;
                        else
                                if (p==ultimo)
                                        ultimo=q;

                }
        }


        return;

}


int main(int argc, char *argv[])
{
        int cant;

        string cod, nombre, c;
        int opc;

        iniciar();

        do
                {

        system("cls");

        cout<<"         LISTA DE ASIGNATURAS"<<endl;
        cout<<"     Seleccione la opcion a realizar\n\n";
        cout<<"     1.- Insertar una asignatura\n";
        cout<<"     2.- Eliminar una asignatura\n";
        cout<<"     3.- Mostrar toda la lista de asignaturas\n";
        cout<<"     4.- Salir";
        cout<<"      \n\n\nOpcion(1-4): ";
```

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

```
            cin>>opc;

            switch(opc)
            {
            case 1:
                  cod="";
                  nombre="";
                  cant=0;
                  cout<<"Entre los Datos de la Asignatura:\n ";
                  cout<<"Codigo o siglas de la Asignatura: \n";
                  cin>>cod;
                  cout<<"Nombre de la Asignatura: \n";
                  cin>>nombre;
                  cout<<"Cantidad de Horas:\n ";
                  cin>>cant;
                  insertar_delante(cod,nombre,cant);
                  break;
            case 2:
                  cout<<"codigo de la Asignatura a eliminar: \n";
                  cin>>c;
                  eliminar(c);
                  getchar();
                  break;
            case 3:
                  mostrar();
                  cout<<"Oprima una tecla para salir";
                  getchar();
                  break;
            case 4:
                  exit(0);
                  break;

            }
      }
      while ((opc!=4));


      return 0;
}
```

EJERCICIOS PARA PRACTICAR

- Para el anterior ejemplo, adicione las siguientes opciones.
  - Insertar al final de la lista.
  - Insertar ordenadamente en la lista.
  - Mostrar la asignatura de mayor cantidad de horas.
  - Mostrar la asignatura con menor cantidad de horas.

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

ORIENTACIÒN PARA LABORATORIO Nro3

- Ejercicio a desarrollar:
  - Implementar la lista de asignaturas, como la vista anteriormente, pero utilizando una lista doblemente enlazada y donde la información de cada nodo es:
    - Código de la asignatura
    - Nombre de la Asignatura.
    - Cantidad de Horas.

- Pasos a seguir:
  - Abra el compilador ZINJAI.
  - Escriba todo el código anterior, o copie y pegue.
  - Realice los cambios pertinentes, tomando en cuenta:
    - Debe implementar sobre una lista doblemente enlazada.
    - Implemente otras operaciones tales como:
      - Buscar una determinada asignatura.
      - Mostrar la asignatura de mayor cantidad de horas.