Links a los Codigos:

- **Header:**
  https://hastebin.com/cuqerorupu.cpp
- **Implementacion:**
  https://hastebin.com/ajejetoqon.cpp
- **Main:**
  https://hastebin.com/osiyipomux.cpp
- **Proyecto:**
  https://drive.google.com/file/d/1bTVyDwMdI53IjqDn9kHoSeS8ZLcCHJLa/view?usp=sharing

Header: TDAListaMemoria

```
//---------------------------------------------------------------------------

#ifndef TDAListaMemoriaH
#define TDAListaMemoriaH
//---------------------------------------------------------------------------
#endif

#include "SMemoria.h"


class TDAListaMemoria
{
    private:
    int Longitud;

  public:
    SMemoria m;
    int PtrElemento;
    // Constructor
    void crear();
    int primero();
    bool vacia();
    int siguiente(int direccion);
    int anterior(int direccion);
    int fin();
    int recupera(int direccion);
    int longitud();
    void inserta(int direccion,int elemento);
    void modifica(int direccion,int elemento);
    void suprime(int direccion);
    void imprimir();
};
```

Implementacion: TDAListaMemoria

```
//---------------------------------------------------------------------------

#pragma hdrstop

#include "TDAListaMemoria.h"
#include <conio.h>
#include <fstream>
#include <iostream>
//---------------------------------------------------------------------------
#pragma package(smart_init)

void TDAListaMemoria::crear()
{
  Longitud = 0;
  PtrElemento = NULO;
}

bool TDAListaMemoria::vacia()
{
    return (Longitud == 0);
}


int TDAListaMemoria::fin()
{
    int PtrFin;
  if(vacia())
  {
        throw("Lista Vacia...");
  }else
  {
    int x = PtrElemento;
    while(x!=NULO)
    {
      PtrFin = x;
      x = m.obtener_dato(x,2);
    }
        return PtrFin;
    }

}

int TDAListaMemoria::primero()
{
  if(!vacia())
  {
    return PtrElemento;
  }else
  {
        throw("Lista Vacia...\n");
    }
}

int TDAListaMemoria::siguiente(int direccion)
{
  if(vacia())
  {
        throw("Lista Vacia...");
  }else
  {
    if(direccion == fin())
    {
      throw("Error, Ultima Direccion...");
    }else
```

```
    {
            return m.obtener_dato(direccion,2);
        }
    }
}

int TDAListaMemoria::anterior(int direccion)
{
  if(vacia())
  {
        throw("Lista Vacia...");
  }else
  {
    if(direccion==primero())
    {
      throw("Error Primera Direccion");
    }else
    {
      int x = PtrElemento;
      int ant = NULO;
      while(x!=NULO)
      {
        if(x==direccion)
        {
                return ant;
        }
        ant = x;
        x = m.obtener_dato(x,2);
            }
        }
  }
}

int TDAListaMemoria::recupera(int direccion)
{
  if(vacia())
  {
        throw("Lista Vacia...");
  }else
  {
        return m.obtener_dato(direccion,1);
  }
}

int TDAListaMemoria::longitud()
{
    return Longitud;
}

void TDAListaMemoria::inserta(int direccion,int elemento)
{
  int x = m.new_espacio(2);

  if(x!=NULO)
  {
    m.poner_dato(x,1,elemento);
    m.poner_dato(x,2,NULO);
    if(vacia())
    {
      PtrElemento = x;
            Longitud = 1;
    }else
    {
      Longitud++;
      if(direccion==primero())
      {
```

```
        m.poner_dato(x,2,direccion);
                PtrElemento = x;
      }else
      {
        int ant = anterior(direccion);
        m.poner_dato(ant,2,x);
                m.poner_dato(x,2,direccion);
            }

        }
  }else
  {
    throw("Existe  Espacio de Memoria...");
    }
}

void TDAListaMemoria::modifica(int direccion,int elemento)
{
  if(vacia())
  {
        throw("Lista esta Vacia...");
  }else
  {
        m.poner_dato(direccion,1,elemento);
    }

}


void TDAListaMemoria::imprimir()
{
  if(vacia())
  {
    throw("Lista Vacia...");
  }else
  {
    int p = primero();
    std::cout<<"<";

    while(p!=fin())
    {
      int e = recupera(p);
      std::cout<<e<< ((p!=NULO)?", ":"") ;
      p = siguiente(p);
    }
        std::cout<<recupera(fin());

    std::cout<<">"<<std::endl;

  }
}


void TDAListaMemoria::suprime(int direccion)
{
  if(Longitud==0)
  {
        throw("Lista Vacia...");
  }
  if(direccion==PtrElemento)
  {
    int x = PtrElemento;
    PtrElemento = m.obtener_dato(PtrElemento,2);
    m.poner_dato(direccion,PtrElemento,NULO);
  }else
  {
```

```
    int ant = anterior(direccion);
    m.poner_dato(ant,2,siguiente(direccion));
        m.poner_dato(direccion,ant,NULO);
    }
}
```

Implementacion: Main del Proyecto 1

```
/*////////////////////////////////////////////////////////////////////////////
  Tarea #3
  Docente: Mario Lopez Winnipeg
  U.A.G.R.M.
  Facultad  de Ingenieria en Ciencias de la Computacion y Telecomunicaciones
  Leonardo Anez Vladimirovna
*////////////////////////////////////////////////////////////////////////////

#include <iostream.h>
#include <conio.h>
#include <fstream>
#include <string>
#include <vector>
#include <cstdlib>

#include "SMemoria.h"
//#include "TDAListaVector.h"
#include "TDAListaMemoria.h"
//#include "TDAListaPuntero.h"


using namespace std;


void MostrarOpciones()
{
  cout<<"*********************************\n";
  cout<<" Lista con Simulacion de Memoria \n";
  cout<<"*********************************\n";
  cout<<"Menu Proyecto 1:\n";
  cout<<"\n";
  cout<<"[1] Crear Memoria\n";
  cout<<"[2] Pedir Espacio\n";
  cout<<"[3] Liberar Espacio\n";
  cout<<"[4] Crear Lista\n";
  cout<<"[5] Poner en Direccion Inicial\n";
  cout<<"[6] Poner en Direccion Fin\n";
  cout<<"[7] Mostrar Memoria\n";
  cout<<"[8] Mostrar Lista\n";
  cout<<"[9] Salir\n";
  cout<<"Opcion: ";
}

int main()
{

    // Declaracion de la Lista (Basada en SMemoria)
  TDAListaMemoria LM;

  int opcion,elemento,cantidad,direccion;

    do
  {
    // Refresh de la Consola
    system("cls");

        // Menu de Opciones
    MostrarOpciones();

    // Opcion del Menu
    cin>>opcion;

    switch(opcion)
    {
```

```
case 1:
  // Constructor de la Memoria...
  LM.m = SMemoria();
  break;
case 2:
  // Pedir espacio en Memoria...
  cout<<"Digite la Cantidad de Memoria: ";

  cin>>cantidad;

  LM.m.new_espacio(cantidad);
  break;
case 3:
          // Liberar espacio en Memoria...
          cout<<"Digite la Direccion de Memoria: ";

  cin>>direccion;

  LM.m.delete_espacio(direccion);
  break;
case 4:
          // Creacion de la Lista...
          LM.crear();
  break;
case 5:
  // Agregacion de elemento en la cabeza de la lista
  cout<<"Ingrese el Elemento a Insertar: ";

  cin>> elemento;

  if(LM.vacia())    // Pequena Excepcion
  {
    LM.inserta(1,elemento);
  }else
  {
             LM.inserta(LM.primero(),elemento);
  }
  break;
case 6:
  // Agregacion de elemento en la cola de la lista
  cout<<"Ingrese el Elemento a Insertar: ";

  cin>>elemento;
          if(LM.vacia())
   {
     LM.inserta(1,elemento);
   }else
   {
    LM.inserta(LM.fin(),elemento);
   }
  break;
case 7:
          // Mostramos la Memoria
  LM.m.mostrar();
  getch();
  break;
case 8:
           // Mostramos la Lista
   if(LM.vacia())
   {
     cout<< "Lista Vacia"<<endl;
   }else
   {
               LM.imprimir();
          }
   getch();
```

```
          break;
       default :
         break;
    }

  }while(opcion!=9);

    cout<< "Gracias por usar el programa..." <<endl;

  getch();
  return 0;
}
```