Programación Orientada a Objetos

Adicionales UNIDAD 3

Aplicaciones de Punteros a Objetos
Estructuras Dinámicas en C++
Actividades

Ejercicios

3.2.1- Implemente en un programa C++ una lista dinámica enlazadas de tipo pila de a través de la clase Pila, donde a a su vez cada nodo a enlazar es un objeto (tipo Nodo) y la información relevante del nodo es un dato de tipo entero. Propon-ga la lista de datos: 25, 35, 45, 55, 65 para ingresarlos en modo consola y que se vayan apilando. Luego elimine los nodos uno a uno.

```
//PILAS en C++
//Prof. Gerardo Sas
//e-fich.unl.edu.ar
#include <iostream>
using namespace std;

class nodo {
private:
        int valor; //Aquí guardamos el dato
        nodo *siguiente; //Apunta al siguiente nodo
public:
        nodo(int v, nodo *sig = NULL) //constructor sobrecargado
        { valor = v; siguiente = sig; }
        friend class pila;//Amiga de la clase pila
};
typedef   nodo *pnodo;

class pila {
private:
        pnodo tope;
public:
        pila() : tope(NULL) {} ;
        ~pila();
        void Push(int v);
        int Pop();
        void Mostrar();};

void pila::Push(int v) {
        pnodo nuevo; /* Crear un nodo nuevo */
        nuevo = new nodo(v, tope);
```

```
        /* Ahora nuevo apunta al tope o extremo de la pila,
        donde se agregó el nuevo nodo */
        tope = nuevo;
}

int pila::Pop() {
        nodo *aux; /* nodo es puntero  auxiliar */
        int v; /* variable auxiliar para retornar el valor agregado */
        if(!tope) return 0; /*Si no hay nodos en la pila retornamos 0 */
        aux = tope; /*  Aux apunta al primer elemento de la pila */
        tope = tope->siguiente; /* Tope apunta al penultimo elemento */
        v = aux->valor; /* Guardamos el valor de retorno */
        delete aux; //Elimino la instacia de la memoria de la clase nodo
        return v;
}

pila::~pila(){ //Destructor elimina todos los nodos
        nodo *aux;
        while(tope)
        {aux= tope;
        tope = tope->siguiente;
        delete aux;
        }
        cout << endl;
}

void pila::Mostrar()
{
        nodo *aux;
        aux= tope;
        int cc=0;
        while(aux)
        {
                cout<<(cc++)+1<<"_("<< aux->valor <<")-> ";
                aux = aux->siguiente;
        }
        cout <<"NULL"<<endl;
}

int main(int argc, char *argv[]) {
        pila p;
        int x;
        p.Push(25);
        p.Push(35);
        p.Push(45);
        p.Push(55);
        p.Push(65);
        p.Mostrar();
        for (x=0; x<5; x++){
```

```
                cout<<endl<<"Pop "<<p.Pop()<<endl;
                p.Mostrar();
        };


        return 0;
}
```

3.2.2- Ídem al ejercicio 1, pero utilizando una estructura de tipo cola. La lista a ingresar para formar la cola es: 12, 18, 22, 16, 30, 34. Luego de ingresar 3 datos elimine uno de la cola; luego de ingresar los otros 3 elimine otro nodo. ¿Cuáles nodos quedan en la cola?

```
#include <iostream>
#include <stdlib.h>
//Prof. Gerardo Sas
//gsas@fich.unl.edu.ar
//e-fich.unl.edu.ar
//COLAS en C++

using namespace std;

class nodo {
private:
        int valor;
        nodo *siguiente;
public:
        nodo(int v, nodo *sig = NULL) {
                valor = v; siguiente = sig; };
        friend class cola;
};

typedef nodo *pnodo;

class cola {
private:
        pnodo primero, ultimo;
public:
        cola() { ultimo=(NULL); primero=(NULL);};
        ~cola();
        void Agregar(int v);
        int Quitar(); // equivale a eliminar
        void Mostrar();
};

cola::~cola() {
        while(primero)  Quitar();
}
```

```
void cola::Agregar(int v)  {
        pnodo nuevo;
        nuevo = new nodo(v); /* Crear un nodo nuevo */
        if(ultimo)
                ultimo->siguiente = nuevo; /* Si la cola no estaba vacía, añadimos el
nuevo a continuación de ultimo */
        if(!primero)
                primero = nuevo;  /* Si primero es NULL, la cola estaba vacía, ahora
primero apuntará también al nuevo nodo */
        ultimo = nuevo; /* Ahora, el último elemento de la cola es el nuevo nodo */
}

int cola::Quitar() { /* Quitar equivale a eliminar el nodo*/
        pnodo aux;  /* puntero auxiliar para manipular nodo */
        int v;      /* variable auxiliar para retorno */
        aux = primero; /* Nodo apunta al primer elemento de la pila */
        if(!aux) return 0; /*Si no hay nodos en la pila retornamos 0 */
        /* Asignamos a primero la dirección del segundo nodo */
        primero = aux->siguiente;
        v = aux->valor; /* Guardamos el valor de retorno */
        delete aux; /* Borrar el nodo */
        if(!primero)
                ultimo = NULL; /* Si la cola quedó vacía, ultimo
        debe ser NULL  ntambién*/
        return v;
}

void cola::Mostrar()
{
        nodo *aux;
        aux= primero;
        cout<<"Primero: "<<primero->valor<<" - Ultimo: "<<ultimo->valor<<endl;
        int cc=0;
        while(aux)
        {
                cout<<" ("<< aux->valor <<")-> ";
                aux = aux->siguiente;
        }
        cout <<"NULL"<<endl;
}

int main(int argc, char *argv[]) {
        cola c;
        cout<<"C O L A S en  C++"<<endl;
        cout<<"\nAgrego 3 numeros"<<endl;
        c.Agregar(12);
        c.Agregar(18);
        c.Agregar(22);
        c.Mostrar();
```

```
        cout<<"\nQuito 1"<<endl;
        c.Quitar();
        c.Mostrar();
        cout<<"\nAgrego 3 mas"<<endl;
        c.Agregar(16);
        c.Agregar(30);
        c.Agregar(34);
        c.Mostrar();
        cout<<"\nQuito 1"<<endl;
        c.Quitar();
        c.Mostrar();
        return 0;
}
```

3.2.3- Proponga una lista dinámica que enlace objetos de tipo Nodo (con un entero entre sus atributos). Diseñe al clase Lista_din con métodos para insertar un nuevo nodo, crear la lista, eliminar un nodo. Proponga los atributos necesarios y otros métodos si lo cree conveniente.

```
//LISTAS DINAMICAS EN C++
//*************************************************
//Prof. Gerardo Sas
//http://e-fich.unl.edu.ar
//*************************************************
#include <iostream>
#include <stdlib.h>
using namespace std;
//*************************************************
class nodo
        {
        private:
                int valor;
                nodo *siguiente;

                friend class Lista_din;

        public:
                nodo(int v, nodo *sig = NULL)
                {
                        valor = v;
                        siguiente = sig;
                }
        };
//*************************************************
typedef nodo *pnodo;
//*************************************************
class Lista_din
        {
        private:
```

```
		pnodo primero;
		pnodo actual;
	public:
		Lista_din();
		~Lista_din();

		void Insertar(int v);
		void Borrar(int v);
		bool ListaVacia() { return primero == NULL; }
		void Mostrar();
		void Siguiente();
		void Primero();
		void Ultimo();
		bool Actual() { return actual != NULL; }
		int ValorActual() { return actual->valor; }
	};
//************************************************
Lista_din::Lista_din(){ primero = actual = NULL; }
Lista_din::~Lista_din(){
	pnodo aux;

	while(primero)
	{
		aux = primero;
		primero = primero->siguiente;
		delete aux;
	}
	actual = NULL;
}

void Lista_din::Insertar(int v)
{
	pnodo anterior;

	// Si la lista está vacía
	if(ListaVacia() || primero->valor > v)
		// Asignamos a lista un nuevo nodo de valor v y
		// cuyo siguiente elemento es la lista actual
		primero = new nodo(v, primero);

	else
		{
			// Buscar el nodo de valor menor a v
			anterior = primero;
			// Avanzamos hasta el último elemento o hasta que el siguien-
te tenga
			// un valor mayor que v
			while(anterior->siguiente && anterior->siguiente->valor <= v)
				anterior = anterior->siguiente;
```

```
                        // Creamos un nuevo nodo después del nodo anterior, y cuyo
siguiente es el si-guiente del anterior
                        anterior->siguiente = new nodo(v, anterior->siguiente);
                }
}

void Lista_din::Borrar(int v)
{
        pnodo anterior, nodo;
        nodo = primero;
        anterior = NULL;
        while(nodo && nodo->valor < v)
        {
                anterior = nodo;
                nodo = nodo->siguiente;
        }
        if(!nodo || nodo->valor != v) return;

        else
                {
                        // Borrar el nodo
                        if(!anterior) // Primer elemento
                                primero = nodo->siguiente;
                        else  // un elemento cualquiera
                                anterior->siguiente = nodo->siguiente;
                        delete nodo;
                }
}

void Lista_din::Mostrar()
{
        nodo *aux;
        aux = primero;
        while(aux)
        {
                cout << aux->valor << "-> ";
                aux = aux->siguiente;
        }
        cout <<"NULL"<< endl;
}

void Lista_din::Siguiente()
{
        if(actual) actual = actual->siguiente;
}

void Lista_din::Primero()
{
        actual = primero;
```

```
}

void Lista_din::Ultimo()
{
        actual = primero;
        if(!ListaVacia())
                while(actual->siguiente) Siguiente();
}


int main(int argc, char *argv[]) {
        Lista_din l;
        cout<<"L I S T A S  D I N A M I C A S  en C++ "<<endl;
        cout<<"\nAgrego 3 numeros"<<endl;
        l.Insertar(12);
        l.Insertar(18);
        l.Insertar(22);
        l.Mostrar();
        cout<<"\nQuito el Num 18"<<endl;
        l.Borrar(18);
        l.Mostrar();
        cout<<"\nAgrego 3 mas"<<endl;
        l.Insertar(16);
        l.Insertar(30);
        l.Insertar(34);
        l.Mostrar();
        cout<<"\nQuito el Num 30"<<endl;
        l.Borrar(30);
        l.Mostrar();
        return 0;
}
```

3.2.4- Como modificaría el tipo Nodo y los métodos de la clase Pila si la información relevante de un nodo es la edad (int) y la altura (float) de un grupo de personas. Proponga los cambios y pruébelos en un programa C++ .

```
//PILAS en C++
#include <iostream>
#include <stdlib.h>
//Prof. Gerardo Sas
//e-fich.unl.edu.ar
using namespace std;

class nodo {
private:
        int edad; float altura; //Aquí guardamos los datos
        nodo *siguiente; //Apunta al siguiente nodo
public:
        nodo(int v, float a, nodo *sig = NULL) //constructor sobrecargado
```

```
        { edad = v; altura = a; siguiente = sig; }
        friend class pila;//Amiga de la clase pila
};
typedef   nodo *pnodo;

class pila {
private:
        pnodo tope;
public:
        pila() : tope(NULL) {} ;
        ~pila();
        void Push(int v, float a);
        int Pop();
        void Mostrar();};

void pila::Push(int v, float a) {
        pnodo nuevo; /* Crear un nodo nuevo */
        nuevo = new nodo(v, a, tope);
        /* Ahora nuevo apunta al tope o extremo de la pila,
        donde se agregó el nuevo nodo */
        tope = nuevo;
};

int pila::Pop() {
        nodo *aux; /* nodo es puntero  auxiliar */
        int v; /* variable auxiliar para retornar el edad agregado */
        if(!tope) return 0; /*Si no hay nodos en la pila retornamos 0 */
        aux = tope; /*  Aux apunta al primer elemento de la pila */
        tope = tope->siguiente; /* Tope apunta al penultimo elemento */
        v = aux->edad; /* Guardamos el edad de retorno */
        delete aux; //Elimino la instacia de la memoria de la clase nodo
        return v;
};

pila::~pila(){ //Destructor elimina todos los nodos
        nodo *aux;
        while(tope)
        {aux= tope;
        tope = tope->siguiente;
        delete aux;
        }
        cout << endl;
}

void pila::Mostrar()
{
        nodo *aux;
        aux= tope;
        int cc=0;
```

```
        while(aux)
        {
                cout<<cc++<<"- Edad: "<< aux->edad <<" Altura: "<<aux-
>altura<<endl;
                aux = aux->siguiente;
        }
        cout << endl;
}

int main(int argc, char *argv[]) {
        pila p;
        int x;
        p.Push(25,1.70);
        p.Push(35,1.71);
        p.Push(45,1.72);
        p.Push(55,1.73);
        p.Push(65,1.74);
        p.Mostrar();
        for (x=0; x<5; x++){
                cout<<"Pop   "<<endl;
                p.Pop();
        };
        p.Mostrar();

        return 0;
}
```

2.3.1. Crear una Aplicación que instancie y utilice la clase pila creada en el ejercicio 3.2.1. Debe presentar un menú para que el usuario pueda Agregar, Quitar y Mostrar los nodos de la Pila.

```
//****** CUERPO PRINCIPAL DEL PROGRAMA ********
void main(void){
  pila P;
  int x, val;
  for (x=0; x<10; x++){
                val= rand()%100;
        P.Push(val);
  };
  P.Mostrar();
int op= -1;
while(op!=0){
        cout<<endl<<"Menu de Opciones"<<endl;
        cout<<" 0 - Salir."<<endl;
        cout<<" 1- Insertar."<<endl;
        cout<<" 2- Borrar."<<endl;
        cout<<"Elija una opcion (0,1,2): ";
        cin>>op;
   cout<<endl<<endl;
```

```
switch( op ) {
   case 0:
     break;
   case 1:
    {int va;
          cout<<"Valor a insertar: ";
          cin>>va;
          P.Push(va);
          P.Mostrar();};
    break;
   case 2:
    {int va;
          //cout<<"Valor a Borrar: ";cin>>va;
          P.Pop();
          P.Mostrar();};
    break;
       default:
     cout<<"ERROR OPCION NO VALIDA";
    break;
  };//del switch
};//del while
}//del main
```

2.3.2. Oriéntese de igual forma que en el actividad anterior modelando una Cola en lugar de una Pila.

```
//C O L A S  en C++
#include <iostream>
#include <stdlib.h>
//Prof. Gerardo Sas
//gsas@fich.unl.edu.ar
//e-fich.unl.edu.ar
using namespace std;

class nodo {
private:
        int valor;
        nodo *siguiente;
public:
        nodo(int v, nodo *sig = NULL) {
                valor = v; siguiente = sig; };
        friend class cola;
};

typedef nodo *pnodo;

class cola {
private:
        pnodo primero, ultimo;
```

```
public:
        cola() { ultimo=(NULL); primero=(NULL);};
        ~cola();
        void Agregar(int v);
        int Leer(); // equivale a eliminar
        void Mostrar();
};

cola::~cola() {
        while(primero)  Leer();
}

void cola::Agregar(int v)  {
        pnodo nuevo;
        nuevo = new nodo(v); /* Crear un nodo nuevo */
        if(ultimo)
                ultimo->siguiente = nuevo; /* Si la cola no estaba vacía, añadimos el
nuevo a continuación de ultimo */
        if(!primero)
                primero = nuevo;  /* Si primero es NULL, la cola estaba vacía, ahora
primero apuntará también al nuevo nodo */
        ultimo = nuevo; /* Ahora, el último elemento de la cola es el nuevo nodo */
}

int cola::Leer() { /* Leer equivale a eliminar el nodo*/
        pnodo aux;  /* puntero auxiliar para manipular nodo */
        int v;      /* variable auxiliar para retorno */
        aux = primero; /* Nodo apunta al primer elemento de la pila */
        if(!aux) return 0; /*Si no hay nodos en la pila retornamos 0 */
        /* Asignamos a primero la dirección del segundo nodo */
        primero = aux->siguiente;
        v = aux->valor; /* Guardamos el valor de retorno */
        delete aux; /* Borrar el nodo */
        if(!primero)
                ultimo = NULL; /* Si la cola quedó vacía, ultimo
        debe ser NULL  ntambién*/
        return v;
}

void cola::Mostrar()
{
        nodo *aux;
        aux= primero;
        while(aux)
        {
                cout<<"("<< aux->valor <<")-> ";
                aux = aux->siguiente;
        }
        cout <<"NULL"<<endl<<endl;
```

```
}

//****** CUERPO PRINCIPAL DEL PROGRAMA ********
int main(){
	cola C;
	int x, val;
	for (x=0; x<10; x++){
		val= rand()%100;
		C.Agregar(val);
	};
	C.Mostrar();
	int op= -1;
	while(op!=0){
		cout<<endl<<"C o l a   e n  C + +\nMenu de Opciones"<<endl;
		cout<<" 0 - Salir."<<endl;
		cout<<" 1- Insertar."<<endl;
		cout<<" 2- Borrar."<<endl;
		cout<<"Elija una opcion (0,1,2): ";
		cin>>op;
		cout<<endl<<endl;
		switch( op ) {
		case 0:
			break;
		case 1:
			{int va;
			cout<<"Valor a insertar: ";
			cin>>va;
			C.Agregar(va);
			C.Mostrar();};
			break;
		case 2:
			{C.Leer();
			C.Mostrar();};
			break;
		default:
			cout<<"ERROR OPCION NO VALIDA";
			break;
		};//del switch
	};//del while
	return 0;
}//del main
```

2.3.3. Modifique alguno de los ejercicios anteriores para que los nodos funcionen como una lista. Agregue en el menú en la opción Eliminar: al activar esta opción la aplicación debe solicitar el valor a eliminar de la lista. Luego debe recorrer la lista hasta hallar el valor ingresado y eliminar el nodo, o bien emitir un mensaje que diga 'Valor no encontrada'.

```
//LISTAS  DINAMICAS  en C++
```

```
//*************************************************
//Prof. Gerardo Sas
//http://e-fich.unl.edu.ar
//*************************************************
#include <iostream>
#include <stdlib.h>
using namespace std;
//*************************************************
class nodo
	{
	private:
		int valor;
		nodo *siguiente;

		friend class lista;

	public:
		nodo(int v, nodo *sig = NULL)
		{
			valor = v;
			siguiente = sig;
		}
	};
//*************************************************
typedef nodo *pnodo;
//*************************************************
class lista
	{
	private:
		pnodo primero;
		pnodo actual;
	public:
		lista() { primero = actual = NULL; }
		~lista();

		void Insertar(int v);
		void Borrar(int v);
		bool ListaVacia() { return primero == NULL; }
		void Mostrar();
		void Siguiente();
		void Primero();
		void Ultimo();
		bool Actual() { return actual != NULL; }
		int ValorActual() { return actual->valor; }
	};
//*************************************************
lista::~lista()
{
	pnodo aux;
```

```
        while(primero)
        {
                aux = primero;
                primero = primero->siguiente;
                delete aux;
        }
        actual = NULL;
}
//*********************************
void lista::Insertar(int v)
{
        pnodo anterior;

        // Si la lista está vacía
        if(ListaVacia() || primero->valor > v)
                // Asignamos a lista un nuevo nodo de valor v y
                // cuyo siguiente elemento es la lista actual
                primero = new nodo(v, primero);

        else
                {
                        // Buscar el nodo de valor menor a v
                        anterior = primero;
                        // Avanzamos hasta el último elemento o hasta que el siguien-
te tenga
                        // un valor mayor que v
                        while(anterior->siguiente && anterior->siguiente->valor <= v)
                                anterior = anterior->siguiente;
                        // Creamos un nuevo nodo después del nodo anterior, y cuyo
siguiente es el si-guiente del anterior
                        anterior->siguiente = new nodo(v, anterior->siguiente);
                }
}

void lista::Borrar(int v)
{
        pnodo anterior, nodo;
        nodo = primero;
        anterior = NULL;
        while(nodo && nodo->valor < v)
        {
                anterior = nodo;
                nodo = nodo->siguiente;
        }
        if(!nodo || nodo->valor != v) return;

        else
                {
```

```
                        // Borrar el nodo
                        if(!anterior) // Primer elemento
                                primero = nodo->siguiente;
                        else  // un elemento cualquiera
                                anterior->siguiente = nodo->siguiente;
                        delete nodo;
                }
}

void lista::Mostrar()
{
        nodo *aux;
        aux = primero;
        while(aux)
        {
                cout << aux->valor << "-> ";
                aux = aux->siguiente;
        }
        cout <<"NULL"<< endl;
}

void lista::Siguiente()
{
        if(actual) actual = actual->siguiente;
}

void lista::Primero()
{
        actual = primero;
}

void lista::Ultimo()
{
        actual = primero;
        if(!ListaVacia())
                while(actual->siguiente) Siguiente();
}
//+++++ CUERPO PRINCIPAL DEL PROGRAMA ++++++++++++
int main(void){
        lista ilista;
        int x, val;
        for (x=0; x<5; x++){
                val= rand()%1000;
                ilista.Insertar(val);
        };
        ilista.Mostrar();
        int op= -1;
        while(op!=0){
```

```
                cout<<endl<<"LISTAS  DINAMICAS  en C++\nMenu de Opcio-
nes"<<endl;
                cout<<" 0 - Salir."<<endl;
                cout<<" 1- Insertar."<<endl;
                cout<<" 2- Borrar."<<endl;
                cout<<"Elija una opcion (0,1,2): ";
                cin>>op;
                cout<<endl<<endl;
                switch( op ) {
                case 0:
                        break;
                case 1:
                        {int va;
                        cout<<"Valor a insertar: ";
                        cin>>va;
                        ilista.Insertar(va);
                        ilista.Mostrar();};
                        break;
                case 2:
                        {int va;
                        cout<<"Valor a Borrar: ";cin>>va;
                        ilista.Borrar(va);
                        ilista.Mostrar();};
                        break;
                default:
                        cout<<"ERROR OPCION NO VALIDA";
                        break;
                };//del switch
        };//del while
        return 0;
}
```