```cpp
#include <iostream>
#include <locale>

using namespace std;
const double PI = 3.141592;

class Figura{
  protected:
    float x;
  public:
    Figura(float cx=0){
       x = cx;
      }
  virtual float Perimetro() = 0;
  virtual float Area() = 0;
};

// Clases derivadas
class Circulo: public Figura{
   public:

   Circulo(float radio){
     x = radio;
   }
   float Perimetro(){
     return 2* PI * x;
   }
   float Area(){
     return PI * x * x;
    }
};

class Cuadrado: public Figura
{

 public:
  Cuadrado(float lado){
    x = lado;
  }
  float Perimetro(){
    return 4 * x;
  }
  float Area(){
    return x * x;
  }
};

int main(){
   setlocale(LC_ALL, "");

   float l, r;
```

```
cout << "Entre el lado del cuadrado" << endl;
cin >> l;

cout << "Entre el radio del círculo" << endl;
cin >> r;

Cuadrado cuad1(l);
Circulo  Circ1(r);

cout << "El perimetro del circulo es:" << Circ1.Perimetro() << endl;
cout << "El área del circulo es:" << Circ1.Area() << endl;

cout << "El perimetro del cuadrado es:" << cuad1.Perimetro() << endl;
cout << "El area del cuadrado es:" << cuad1.Area() << endl;
}
```

Imagen 1.Encapsulamiento y Herencia.

```
#include <iostream>
using namespace std;

class pareja {
   public:
      // Constructor
      pareja(int a2, int b2);
      // Funciones miembro de la clase "pareja"
      void Lee(int &a2, int &b2);
      void Guarda(int a2, int b2);
   private:
      // Datos miembro de la clase "pareja"
      int a, b;
};

pareja::pareja(int a2, int b2) {
   a = a2;
   b = b2;
}

void pareja::Lee(int &a2, int &b2) {
   a2 = a;
   b2 = b;
}

void pareja::Guarda(int a2, int b2) {
   a = a2;
   b = b2;
}

int main() {
   pareja par1(12, 32);
   int x, y;

   par1.Lee(x, y);
   cout << "Valor de par1.a: " << x << endl;
   cout << "Valor de par1.b: " << y << endl;

   return 0;
}
```

Imagen 2.Constructor de la clase pareja.

```cpp
#include <iostream>
using namespace std;

class Tiempo {
  public:
   Tiempo(int h=0, int m=0) : hora(h), minuto(m) {}

   void Mostrar();
   Tiempo operator+(Tiempo h);

  private:
   int hora;
   int minuto;
};

Tiempo Tiempo::operator+(Tiempo h) {
   Tiempo temp;

   temp.minuto = minuto + h.minuto;
   temp.hora   = hora   + h.hora;

   if(temp.minuto >= 60) {
      temp.minuto -= 60;
      temp.hora++;
   }
   return temp;
}

void Tiempo::Mostrar() {
   cout << hora << ":" << minuto << endl;
}

int main() {
   Tiempo Ahora(12,24), T1(4,45);

   T1 = Ahora + T1;   // (1)
   T1.Mostrar();

   (Ahora + Tiempo(4,45)).Mostrar(); // (2)

   return 0;
}
```

Imagen 3.Operadores.

```cpp
#include <iostream>
using namespace std;

class ClaseA {
  public:
   ClaseA() : datoA(10) {}
   int LeerA() const { return datoA; }
   void Mostrar() {
      cout << "a = " << datoA << endl; // (1)
   }
  protected:
   int datoA;
};

class ClaseB : public ClaseA {
  public:
   ClaseB() : datoB(20) {}
   int LeerB() const { return datoB; }
   void Mostrar() {
      cout << "a = " << datoA << ", b = "
           << datoB << endl; // (2)
   }
  protected:
   int datoB;
};

int main() {
   ClaseB objeto;

   objeto.Mostrar();
   objeto.ClaseA::Mostrar();

   return 0;
}
```

Imagen 4.Funciones.

```
#include <iostream>
using namespace std;

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

   private:
    int valor;
    nodo *siguiente;

   friend class lista;
};

typedef nodo *pnodo;

class lista {
   public:
    lista() { primero = actual = NULL; }
    ~lista();

    void Insertar(int v);
    void Borrar(int v);
    bool ListaVacia() { return primero == NULL; }
    void Mostrar();
    void Siguiente() { if(actual) actual = actual->siguiente; }
    void Primero() { actual = primero; }
    void Ultimo() { Primero(); if(!ListaVacia()) while(actual->siguiente) Siguiente(); }
    bool Actual() { return actual != NULL; }
   int ValorActual() { return actual->valor; }

  private:
   pnodo primero;
   pnodo actual;
};

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 siguiente tenga
      // un valor mayor que v
      while(anterior->siguiente && anterior->siguiente->valor <= v)
         anterior = anterior->siguiente;
```

Imagen 5.Estructuras de datos. Parte 1

```
        // Creamos un nuevo nodo después del nodo anterior, y cuyo siguiente
        // es el siguiente 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 << endl;
}
int main() {
    lista Lista;

    Lista.Insertar(20);
    Lista.Insertar(10);
    Lista.Insertar(40);
    Lista.Insertar(30);
    Lista.Mostrar();
    cout << "Lista de elementos:" << endl;
    Lista.Primero();
    while(Lista.Actual()) {
        cout << Lista.ValorActual() << endl;
        Lista.Siguiente();
    }
    Lista.Primero();
    cout << "Primero: " << Lista.ValorActual() << endl;

    Lista.Ultimo();
    cout << "Ultimo: " << Lista.ValorActual() << endl;

    Lista.Borrar(10);
    Lista.Borrar(15);
    Lista.Borrar(45);
    Lista.Borrar(30);
    Lista.Borrar(40);
    Lista.Mostrar();

    return 0;
}
```

Imagen 5.Estructuras de datos. Parte 2

Bibliografía:

[Imagen 1]: LabSCN –*Programación Orientada a objetos en C++*. [Disponible en línea]: https://labscn-unalmed.github.io/programacion-R/clases/clase-03.html [Acceso 2 de septiembre 2020].

[Imagen 2]: Con Clase –*Curso de C++*. [Disponible en línea]: http://c.conclase.net/curso/?cap=029#inicio [Acceso 2 de septiembre 2020].

[Imagen 3]: Con Clase –*Curso de C++*. [Disponible en línea]: http://c.conclase.net/curso/?cap=035#inicio [Acceso 2 de septiembre 2020].

[Imagen 4]: Con Clase –*Curso de C++*. [Disponible en línea]: http://c.conclase.net/curso/?cap=037 [Acceso 2 de septiembre 2020].

[Imagen 5](Parte 1 y Parte 2): Con Clase –*Curso de C++*. [Disponible en línea]: http://c.conclase.net/edd/?cap=001g [Acceso 2 de septiembre 2020].