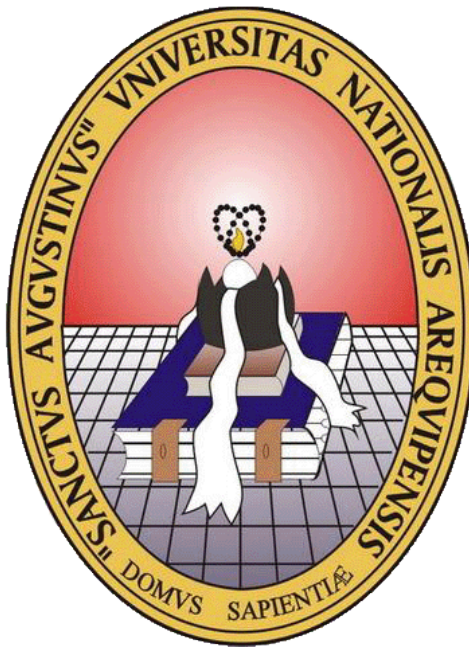


UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA  
FACULTAD DE PRODUCCIÓN Y SERVICIOS  
ESCUELA DE CIENCIAS DE LA COMPUTACIÓN



PRÁCTICA DE LABORATORIO 20  
CURSO DE CIENCIAS DE LA COMPUTACIÓN II

ESTUDIANTE:  
RUIZ MAMANI, EDUARDO GERMÁN

EMAIL: [eruizm@unsa.edu.pe](mailto:eruizm@unsa.edu.pe)

CUI: 20193061

TURNO:

C

AREQUIPA- PERÚ

2021

LINK DEL REPOSITORIO: [https://github.com/EGRM23/CCII\\_20193061.git](https://github.com/EGRM23/CCII_20193061.git)

## 1. EJERCICIO

1. Implemente el siguiente código que usa punteros sin procesar y explique lo que hace:

```
{
    double* d = new double(1.0);
    Point* pt = new Point(1.0, 2.0);

    *d = 2.0;
    (*pt).X(3.0);
    (*pt).Y(3.0);

    pt->X(3.0);
    pt->Y(3.0);

    delete d;
    delete pt;
}
```

- **CÓDIGO**

```
{
    //Declaramos un puntero de tipo double y lo inicializamos
    con valor 1.0
    double* d = new double(1.0);

    //Declaramos un puntero de tipo Point y lo inicializamos
    con 1,0 y 2.0
    Point* pt = new Point(1.0, 2.0);

    //Cambiamos el valor del puntero d a 2.00
    *d = 2.0;

    //Cambiamos el valor de X mediante un método de la clase
    point, usamos el
    //valor al que apunta el puntero
    (*pt).X(3.0);

    //Cambiamos el valor de Y mediante un método de la clase
    point, usamos el
    //valor al que apunta el puntero
    (*pt).Y(3.0);

    //Aquí también cambiamos el valor de X, pero usamos el
    puntero directamente
    //con ->
    pt->X(3.0);

    //Aquí también cambiamos el valor de Y, pero usamos el
    puntero directamente
    //con ->
    pt->Y(3.0);

    //Eliminamos los punteros creados
    delete d;
```

```

        delete pt;
    }

```

## 2. EJERCICIO

- Transfiera el código anterior reemplazando los punteros sin formato por `std::unique_ptr`.

- **CÓDIGO**

```

{
    std::unique_ptr<double> d = std::make_unique<double>(1.0);
    std::unique_ptr<Point> pt = std::make_unique<Point>(1.0,
2.0);
    *d = 2.0;
    (*pt).X(3.0);
    (*pt).Y(3.0);

    pt->X(3.0);
    pt->Y(3.0);
}

```

## 3. EJERCICIO

- Implementar el código para las clases C1 y C2, cada una de las cuales contiene el objeto compartido d anterior, por ejemplo:

```

class C1
{
private:
    std::shared_ptr<double> d;
public:
    C1(std::shared_ptr<double> value) : d(value) {}
    virtual ~C1() { cout << "\nC1 destructor\n"; }
    void print() const { cout << "Valor " << *d; }
};

```

- **CÓDIGO:**

```

#include <iostream>
#include <memory>

class C1 {
private:
    std::shared_ptr<double> d;
public:
    C1(std::shared_ptr<double> value) : d(value) {}
    virtual ~C1() { std::cout << "\nC1 destructor\n"; }
    void print() const { std::cout << "C1 Valor " << *d <<
std::endl; }
};

class C2 {
private:
    std::shared_ptr<double> d;
public:
    C2(std::shared_ptr<double> value) : d(value) {}
    virtual ~C2() { std::cout << "\nC2 destructor\n"; }
}

```

```

        void print() const { std::cout << "C2 Valor " << *d <<
std::endl; }
};

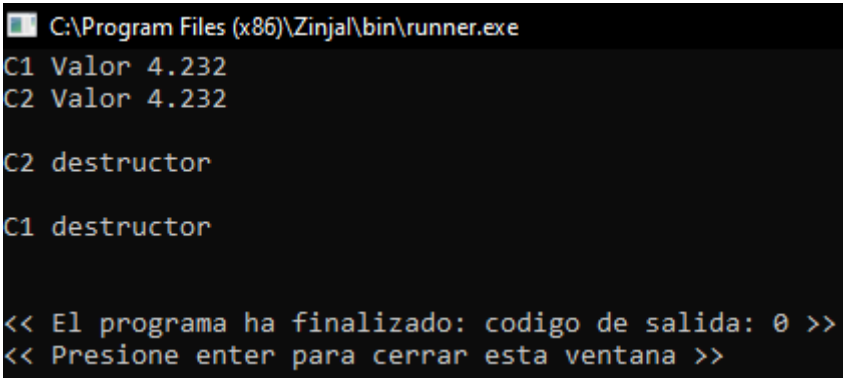
int main() {
    std::shared_ptr<double> dm =
std::make_shared<double>(4.232);

    C1 prueba_c1(dm);
    prueba_c1.print();
    C2 prueba_c2(dm);
    prueba_c2.print();

    return 0;
}

```

- **CAPTURAS**



```

C:\Program Files (x86)\Zinjal\bin\runner.exe
C1 Valor 4.232
C2 Valor 4.232

C2 destructor

C1 destructor

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```

#### 4. EJERCICIO

4. Transfiera el código anterior reemplazando los punteros sin formato por `std::shared_ptr<Point> p;`

- **CÓDIGO:**

```

#include <iostream>
#include <memory>

class Point {
private:
    double x, y;
public:
    Point(double valx, double valy) : x(valx),y(valy) {}
    ~Point() {}
    void print() {
        std::cout << "Valor X: " << this->x << std::endl;
        std::cout << "Valor Y: " << this->y << std::endl;
    }
};

class C1 {
private:
    std::shared_ptr<Point> pt;
public:

```

```

        C1(std::shared_ptr<Point> value) : pt(value) {}
        virtual ~C1() { std::cout << "\nC1 destructor\n"; }
        void print() const {
            std::cout << "CLASS C1\n";
            (*pt).print();
        }
    };

class C2 {
private:
    std::shared_ptr<Point> pt;
public:
    C2(std::shared_ptr<Point> value) : pt(value) {}
    virtual ~C2() { std::cout << "\nC2 destructor\n"; }
    void print() const {
        std::cout << "CLASS C1\n";
        (*pt).print();
    }
};

int main() {
    std::shared_ptr<Point> pm =
std::make_shared<Point>(3.19507,5.11714);

    C1 prueba_c1(pm);
    prueba_c1.print();
    C2 prueba_c2(pm);
    prueba_c2.print();

    return 0;
}

```

- **CAPTURAS**

```

C:\Program Files (x86)\Zinjal\bin\runner.exe
CLASS C1
Valor X: 3.19507
Valor Y: 5.11714
CLASS C1
Valor X: 3.19507
Valor Y: 5.11714

C2 destructor

C1 destructor

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```

## 5. EJERCICIO

5. Al anterior código implemente un puntero débil a un puntero el cual no puede estar vacío.

- **CÓDIGO:**

```
#include <iostream>
#include <memory>

class Point {
private:
    double x, y;
public:
    Point(double valx, double valy) : x(valx),y(valy) {}
    ~Point() {}
    void print() {
        std::cout << "Valor X: " << this->x << std::endl;
        std::cout << "Valor Y: " << this->y << std::endl;
    }
};

class C1 {
private:
    std::weak_ptr<Point> pt;
public:
    C1(std::shared_ptr<Point> value) : pt(value) {}
    virtual ~C1() { std::cout << "\nC1 destructor\n"; }
    void print() {
        std::shared_ptr<Point> temp_pt = pt.lock();
        std::cout << "CLASS C1\n";
        (*temp_pt).print();
    }
};

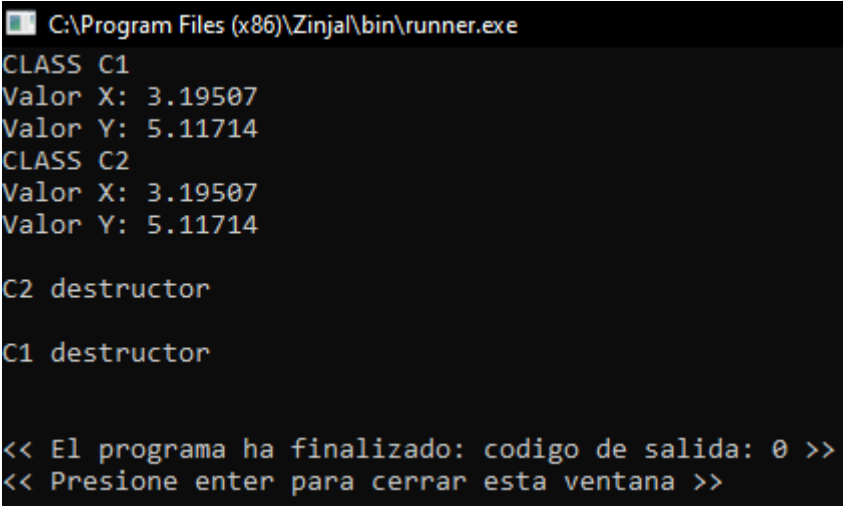
class C2 {
private:
    std::weak_ptr<Point> pt;
public:
    C2(std::shared_ptr<Point> value) : pt(value) {}
    virtual ~C2() { std::cout << "\nC2 destructor\n"; }
    void print() {
        std::shared_ptr<Point> temp_pt = pt.lock();
        std::cout << "CLASS C2\n";
        (*temp_pt).print();
    }
};

int main() {
    std::shared_ptr<Point> pm =
    std::make_shared<Point>(3.19507,5.11714);

    C1 prueba_c1(pm);
    prueba_c1.print();
    C2 prueba_c2(pm);
    prueba_c2.print();
}
```

```
        return 0;  
    }
```

- **CAPTURAS**



```
C:\Program Files (x86)\Zinjal\bin\runner.exe  
CLASS C1  
Valor X: 3.19507  
Valor Y: 5.11714  
CLASS C2  
Valor X: 3.19507  
Valor Y: 5.11714  
  
C2 destructor  
  
C1 destructor  
  
<< El programa ha finalizado: codigo de salida: 0 >>  
<< Presione enter para cerrar esta ventana >>
```