

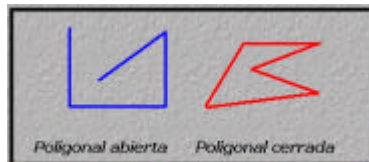
**DII**

Asignatura: **LS1158 Metodología y Técnicas de la Programación II**  
Cuatrimestre: **2º** Examen: **Final** Convocatoria: **Ordinaria**  
Grupo: **1T2-1T3** Curso: **2004/05** Fecha: **31/V/05**

1. (5 Puntos) Dada la clase Punto con la siguiente interfaz ya implementada

```
class Punto {
    float x,y;
public:
    Punto(float = 0, float = 0);
    void Ver() const;           // Visualiza (x,y) en la salida estandar
    void Leer();               // Lee las coordenadas de la entrada estandar
    void CambiarX(float);      // Cambia la coordenada x
    void CambiarY(float);      // Cambia la coordenada y
    void Cambiar(float, float); // Cambia ambas coordenadas
    void Trasladar(float, float); // Suma un desplazamiento a cada coordenada
    float X() const;          // Devuelve la coordenada X
    float Y() const;          // Devuelva la coordenada Y
    float Distancia (Punto) const; // Devuelve la distancia entre dos puntos
};
```

definir una clase Poligonal para representar poligonales con un número arbitrario n de puntos. Recordemos que una Poligonal es una línea formada por segmentos de recta unidos, como por ejemplo:



La clase ha de contener las siguientes implementaciones:

- 1) Constructor con un parámetro entero que indica el número de puntos de la Poligonal. Este constructor inicia todos los atributos de la clase a valores consistentes.
- 2) Constructor sin argumentos. ¿Es necesario implementar este constructor? ¿Es éste el constructor de oficio?

- 3) Constructor copia y destructor. ¿Son necesarios implementarlos?
- 4) Los siguientes operadores sobrecargados:
  - 1) Operador `[]`, el cual devuelve el `Punto` de la poligonal que está en la posición que se indica.
  - 2) Operador `+`, que permita realizar la operación `Poligonal + Punto` devolviendo como resultado una nueva `Poligonal`. Esta nueva `Poligonal` es el resultado de añadir a la `Poligonal` inicial el segmento que une su último punto con el punto con el cual se está sumando.
  - 3) Operador `--` (post-decremento). Elimina el último segmento de la poligonal (tomando como orden el orden de almacenamiento).
  - 4) Operador `+`, que permita realizar la operación `Punto + Poligonal` devolviendo como resultado una nueva `Poligonal`. Esta nueva `Poligonal` es el resultado de añadir a la `Poligonal` inicial el segmento que une su último punto con el punto con el cual se está sumando.
  - 5) Operador `<<` muestra el contenido de la poligonal de la siguiente forma: `{n_seg;(x1,y1),(x2,y2)...(xn,yn)}`. Donde `n_seg` es el número de segmentos que forman la poligonal y seguidamente se muestran la lista de puntos que forma la poligonal.
  - 6) Operador `>>`. Se pide por pantalla en número de segmentos de la poligonal que vamos a insertar. Después se pedirán los datos de los puntos necesarios para formar una poligonal del número de segmentos indicado.

Nota: Cuando sea posible que los operadores puedan actuar sobre el resultado dado por otro operador hay que implementarlo OBLIGATORIAMENTE.

2. Dadas las siguientes clases en C++:

```
class A{
    protected:
        int a;
    public:
        A(){a=0;};
        ~A(){};
        virtual void f(){cout<<"Clase A"<<" "<<a<<endl;}
};

class B:public A{
    private:
        int b;
    public:
        B():A(){b=0;};
        ~B(){};
        void f(){cout<<"Clase B"<<" "<<a<<" "<<b<<endl;}
        void g(){};
};
```

Y el siguiente programa principal:

```
main(){
    A* var1;
    B var3=B();
    var1=&var3;
    var1->f();
}
```

Responda a las siguientes preguntas:

- ¿Es correcto el programa principal (main())? Explica brevemente la respuesta.
- ¿Qué aparece por pantalla al ejecutarse el main()? ¿A qué clase pertenece la función f() que se ejecuta? Explica brevemente la respuesta. ¿Qué cambios han de realizarse en las clases o el programa principal para que se ejecutase la f() de la otra clase? Justifica la respuesta.

c) En el código introducimos el siguiente cambio:

```
class A{
    private:
        int a;
    public:
        A(){a=0;};
        ~A(){};
        virtual void f(){cout<<"Clase A"<<" "<<a<<endl;}
};
```

¿Que cambios hay que realizar en el código de las clases para que todo funcione correctamente?

3. Dadas las siguientes clases en C++:

```
class A{
    protected:
        int a;
    public:
        A(){a=0;};
        ~A(){};
        void f(){cout<<"Clase A"<<" "<<a<<endl;}
        virtual void g()=0;
};

class B:public A{
    private:
        int b;
    public:
        B():A(){b=0;};
        ~B(){};
        void f(){cout<<"Clase B"<<" "<<a<<" "<<b<<endl;}
        void g(){cout<<"g de Clase B"<<endl;};
};
```

Responda a las siguientes preguntas:

- a) Crear un array de punteros de la clase A de tamaño 5 e inicializarlo correctamente con elementos de la clase B.
- b) Utilizar el array anterior para aplicar a los cinco elementos la función g(). ¿Qué se muestra por pantalla?
- c) Utilizar el array anterior para aplicar a los cinco elementos la función f(). ¿Qué se muestra por pantalla?