Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Departamento de Sistemas e Informática
Escuela de Electrónica
Informática Aplicada

Prácticas Nº 4 y 5– Programación orientada a objetos en C++

Problema 1

Diseñe e implemente una clase `Point3D` para representar puntos del espacio y pruébela en una función `main`.

Point3D:

Campos:

Tres campos privados representando sus coordenadas: `x`, `y`, y `z`.

Constructores:

- `Point3D()`: Asigna los valores `(0, 0, 0)` al punto.
- `Point3D(int x1, int y1, int z1)`.
- `Point3D(int low, int high)`: Genera aleatoriamente las coordenadas en el rango dado por los argumentos.

Acceso y actualización: (6 en total)
Métodos de utilidad (públicos):

- `boolean equals(Point3D q)`.
- `boolean equals(int x1, int y1, int z1)`.
- `void shiftX(int tx)`: desplaza coordenada `x` en `tx`.
- `void shiftY(int ty)`: desplaza la coordenada `y` en `ty`.
- `void shiftZ(int tz)`: ídem para la `z`.
- `Point3D translate(int tx, int ty, int tz)`: desplaza el punto.
- `double distanceOrigin()`: retorna la distancia al origen.
- `double distanceToPoint(Point3D q)`: retorna la distancia al punto `q`.
- `int scalarProduct(Point3D q)`: retorna el producto escalar del vector del origen al punto `q` con el vector formado por el punto actual: `(x1*x2 + y1*y2 + z1*z2)`.
- `string whichSpace()`: Retorna un string que indica si el punto está en el semiplano derecho o izquierdo, arriba o abajo y adelante o atrás o si es el origen. Si una coordenada es 0 puede pertenecer a cualquiera de los dos semiespacios.
- `boolean inBox (int xbox, int ybox, int zbox, int width, int height, int depth)`: retorna verdadero si el punto está contenido en una caja definida por el por el vértice superior, izquierdo al frente `(xbox, ybox, zbox)` y de un ancho, altura y profundidad.
- `boolean inSphere (Point3D center, int radius)`: retorna true si el punto actual se encuentra contenido en la esfera determinada por su centro y radio.

Debe definir cualquier constante, variable o método que necesite para hacer lo siguiente en `main()`:

- Declare referencias `Point3D (pt1, pt2, pt3)`.

- Instancie cada punto así: `pt1` inicializado con el primer constructor, `pt2` con el segundo y parámetros (10, -5, 4), pt3 inicializado con el tercero y parámetros `(5, 15)`.
- Muestre los 3 puntos con un breve mensaje.
- Cambie las coordenadas del punto `pt2` a `(9, 8, -6)` y muestre por pantalla.
- Desplace `pt1` en `3`, `-3`, `7` usando los shifts separados para cada coordenada, muestre por pantalla y vea si el punto actualizado es igual a `pt2`, usando el primer método de igualdad.
- Desplace el `pt1` en `6`, `11`, `-13` usando `translate(…)` y generando un nuevo punto `pt4` , muestre el nuevo `pt4`, chequee si `pt4` es igual a `pt2`, usando el segundo método de igualdad.
- Calcule y muestre la distancia de `pt3` al origen.
- Calcule y muestre la distancia entre `pt1` y `pt3`.
- Calcule y muestre el producto escalar de `pt1` y `pt2`.
- Muestre en qué parte del espacio están `pt1`, `pt2` y `pt3`.
- Chequee e imprima si `pt2` está contenido en una caja de vértice `(-12, 20, 8)` y ancho, alto y profundidad de `24`, `40` y `35` respectivamente.
- Chequee e imprima si `pt1` está en una esfera con centro `(1, 2, 5)` y radio `50`.
- Imprima todos los campos de los tres puntos.

Solución:

```
// archivo Point3D.h
using std::string;

class Point3D {
	int x;
	int y;
	int z;

public:
	Point3D (int x1, int y1, int z1): x(x1), y(y1), z(z1) {}
	Point3D (): x(0), y(0), z(0) {}
	Point3D (int low, int high);

	int getX() const;
	int getY() const;
	int getZ() const;

	void setX (int tx);
	void setY (int ty);
	void setZ (int tz);

	//string toString() const;
	bool equals(Point3D q) const;
	bool equals(int x1, int y1, int z1) const;
	void shiftX(int tx);
	void shiftY(int ty);
	void shiftZ(int tz);
	void translate (int tx, int ty, int tz);
	double distanceOrigin() const;
	double distanceToPoint(Point3D q) const;
	int scalarProduct(Point3D q) const;
	string Point3D::whichSpace() const;
	bool Point3D::inBox (int xbox, int ybox, int zbox,
		int width, int height, int depth) const;
	bool inSphere(Point3D center, int radius) const;
};
```

```cpp
// archivo Point3D.cpp
#include <iostream>
#include <sstream>
#include "Point3D.h"

using std::cout;
using std::endl;
using std::stringstream
;

Point3D::Point3D (int low, int high) {
	x = low + int((rand() / RAND_MAX)) * (high - low);
	y = low + int((rand() / RAND_MAX)) * (high - low);
	z = low + int((rand() / RAND_MAX)) * (high - low);
}

int Point3D::getX() const { return x; }
int Point3D::getY() const { return y; }
int Point3D::getZ() const { return z; }

void Point3D::setX (int tx) { x = tx; }
void Point3D::setY (int ty) { y = ty; }
void Point3D::setZ (int tz) { z = tz; }

bool Point3D::equals(Point3D q) const {
	return (x == q.x && y == q.y && z == q.z);
}

bool Point3D::equals(int x1, int y1, int z1) const {
	return (x == x1 && y == y1 && z == z1);
}

void Point3D::shiftX(int tx) { x += tx; }
void Point3D::shiftY(int ty) { y += ty; }
void Point3D::shiftZ(int tz) { z += tz; }
void Point3D::translate (int tx, int ty, int tz) {
	x += tx;
	y += ty;
	z += tz;
}

double Point3D::distanceOrigin() const {
	return sqrt(pow((double)x, 2) + pow((double)y, 2) + pow((double)z, 2));
}

double Point3D::distanceToPoint(Point3D q) const {
	return sqrt(pow(double(x - q.x), 2) + pow(double(y - q.y), 2) + pow(double(z - q.z), 2));
}


int Point3D::scalarProduct(Point3D q) const {
	return x * q.x + y * q.y + z * q.z;
}

string Point3D::whichSpace() const { // 8 'espacios': left/right, up/low, far/back
	// si está en el borde cualquier lado está bien
	string message = "Este punto esta: ";

	if (equals(0, 0, 0)) //  origen
		message = "El punto es el origen!";
	else {
		message += "en el ";
		if (x >= 0) // derecha
```

```
                message += "semiplano derecho, ";
        else
                message += "semiplano izquierdo, ";
        if (y >= 0) // arriba
                message += "superior, ";
        else
                message += "inferior, ";
        if (z >= 0) // delante
                message += "y delante.";
        else
                message += "y detrás.";
    }
    return message;
}

bool Point3D::inBox (int xbox, int ybox, int zbox,
    int width, int height, int depth) const {
// xbox,ybox,zbox: borde superior izquierdo de la caja
    return (x >= xbox) && (x <= xbox + width) // rango de x
                && (y <= ybox) && (y >= ybox - height) // rango de y
                && (z <= zbox) && (z >= zbox - depth); // rango de z
}

bool Point3D::inSphere(Point3D center, int radius) const {
// chequea si este punto está dentro de una esfera de
// centro (cx,cy,cz)y radio=radius
    return distanceToPoint(center) <= radius;
}

int main() {
    int a1 = 10, b1 = -5, c1 = 4; // para constructor
    int a2 = 9, b2 = 8, c2 = -6; // para actualizar coordenadas
    int shift1X = 3,
        shift1Y = -3,
        shift1Z = 7,
        shift2X = 6,
        shift2Y = 11,
        shift2Z = -13;
        // para mover
    int low = 5, high = 15; // para constructor
    int xBox = -12, yBox = 20, zBox = 8;
    // borde superior izquierdo y delante de una caja
    int width = 24, height = 40, depth = 35; // de la caja
    int xCenterSphere = 1, yCenterSphere = 2, zCenterSphere = 5;
    // centro de esfera
    int radius = 50; // radio de esfera

    // declara referencia a puntos 3D
    Point3D pt1, pt2(a1, b1, c1), pt3(low, high);

    // imprime
    cout << "Punto 1 es: (" << pt1.getX() << ", " << pt1.getY() << ", " << pt1.getZ() << ")" <<
endl;
    cout << "Punto 2 es: (" << pt2.getX() << ", " << pt2.getY() << ", " << pt2.getZ() << ")" <<
endl;
    cout << "Punto 3 es: (" << pt3.getX() << ", " << pt3.getY() << ", " << pt3.getZ() << ")" <<
endl;

    // imprime coordenadas de punto 3
    printf("\n\nLas coordenadas del punto 3 son: %d, %d, %d", pt3.getX(), pt3.getY(), pt3.getZ());

    // cambia coordenadas de pt2
    pt2.setX(a2);
```

```
        pt2.setY(b2);
        pt2.setZ(c2);

        cout << endl << "Punto 2 ahora es: ( " << pt2.getX() << ", " << pt2.getY() << ", " <<
pt2.getZ() << ")" << endl;

        // mueve pt1 y chequea si es igual a pt2
        pt1.shiftX(shift1X);
        pt1.shiftY(shift1Y);
        pt1.shiftZ(shift1Z);

        cout << "Punto 1 ahora es: ( "<< pt1.getX() << ", " << pt1.getY() << ", " << pt1.getZ() << ")"
<< endl;
        cout << "Es pt1 igual a pt2? --> "<< pt1.equals(pt2) << endl;

        // mueve pt1 de nuevo y chequea si es igual a pt2
        pt1.translate(shift2X, shift2Y, shift2Z);
        Point3D pt4(pt1);
        cout << endl << "Punto 1 trasladado, ahora pt4 es: ( " << pt1.getX() << ", " << pt1.getY() <<
", " << pt1.getZ() << ")" << endl;
        cout << "Es este nuevo punto transladado pt4 igual a pt2? --> " << pt4.equals(pt2.getX(),
pt2.getY(), pt2.getZ()) << endl;

        // distancia de pt3 al origen
        cout << "Distancia de pt3 al origen es: " << pt3.distanceOrigin() << endl;
        // distancia pt1 -  pt3
        cout << "Distancia entre pt1 y pt3 es: " << pt1.distanceToPoint(pt3) << endl;
        // producto escalar de vectores pt1 , pt2
        cout << "Producto escalar de pt1, pt2 es: " << pt1.scalarProduct(pt2) << endl;
        // en qué espacios están pt1, pt2, pt3?

        cout << "Para pt1 --> " << pt1.whichSpace() << endl;
        cout << "Para pt2 --> " << pt2.whichSpace() << endl;
        cout << "Para pt3 --> " << pt3.whichSpace() << endl;
        // está pt2  en una caja?
        cout << "Está pt2 en la caja dada? --> " << pt2.inBox(xBox, yBox, zBox, width, height, depth)
<< endl;
        // está pt1 en la esfera?
        Point3D center(xCenterSphere, yCenterSphere, zCenterSphere);
        cout << "Está pt1 dentro de la esfera? ---> " << pt1.inSphere(center, radius) << endl;



// Imprimir todos los puntos
        cout << endl << "Punto 1 es ( " << pt1.getX() << ", " << pt1.getY() << ", " << pt1.getZ() <<
")" << endl;
        cout << endl << "Punto 2 es ( " << pt2.getX() << ", " << pt2.getY() << ", " << pt2.getZ() <<
")" << endl;
        cout << endl << "Punto 3 es ( " << pt3.getX() << ", " << pt3.getY() << ", " << pt3.getZ() <<
")" << endl;
}
```

Problema 2

a) En una carrera de velocidad participan un cierto número de atletas, cada uno de ellos tiene un nombre, número, nacionalidad y el tiempo que le tomó correr la carrera. Cree una clase `Atleta` que represente a un atleta. Para ensayar esta clase, una función `main()`, que haga lo siguiente: provea información acerca de dos atletas, cree dos objetos `Atleta` e imprima el nombre del más rápido.

b) Una carrera cubre una cierta distancia y cada carrera tiene un ganador. Cree una clase `Carrera` que maneje esta información. Ajuste la función `main` para crear un objeto de tipo `Carrera` y registre el nombre del atleta más rápido de la carrera.

c) Suponga que desea tener acceso a más información acerca del ganador de la carrera (por ejemplo, el tiempo que tardó o su nacionalidad). No podemos hacer esto desde el objeto `carrera` directamente. Necesitamos cambiar la definición de la clase tal que en lugar de almacenar el nombre del ganador, almacene un objeto `Atleta`. Haga este cambio muestre todos los detalles del ganador de una carrera.

d) Ahora deseamos registrar la información de todos los participantes en la carrera, podemos hacer esto añadiendo un atributo `competidores` a la clase `Carrera`, que será un array de `atletas`. Edite la función que determina el ganador de tal forma que ahora lo haga mirando los tiempos de cada competidor. Pruebe estos cambios.

e) Agregue a la clase la sobrecarga del operador `<` que retorna `true` si el atleta fue más rápido que el atleta pasado como argumento.

f) Vuelva a editar la función que determina el ganador utilizando el operador sobrecargado.

Problema 3

a) Un vagón de un tren tiene 40 asientos, cada uno de ellos puede estar ocupado o vacante. El vagón puede ser de primera o segunda clase. Cree una clase `Carriage` para representar esta información. En el constructor se supondrá que todos los asientos inicialmente están vacantes. Escriba los métodos apropiados de acceso y actualización y un método que vaya ocupando los asientos de la siguiente forma: si el vagón es de primera hay un 10% de probabilidad que los asientos sean ocupados; si es de segunda clase hay un 40% de probabilidad que los asientos sean ocupados. Escriba una función `main()` que contenga un objeto `Carriage`, llénelo aleatoriamente e imprima el estado de cada asiento.

b) Un tren consta de un cierto número de vagones, tiene una estación de partida y una de llegada, un cierto precio para los tickets de primera y otro para los de segunda. Cree una clase `Train` que contenga esta información. Añada una función que llene los asientos de los vagones aleatoriamiente. Cree un método que calcule el total de ventas de tickets. Ajuste su función `main` para probar lo hecho.

Solución:

```
#include <iostream>

using std::cout;
using std::endl;

class Carriage {
	bool asientos [40];
	bool primera;
public:
	Carriage() {};
	Carriage(bool prim);
	bool getAsiento(int ind) { return asientos[ind]; }
	bool getPrimera() { return primera; }
	void setPrimera(bool p) { primera = p; }
```

```
	void ocupar();
};

Carriage::Carriage(bool prim) {
	primera = prim;
	for(int i = 0; i < 40; i++) asientos[i] = 0;
}

void Carriage::ocupar() {
	double probabilidad = primera ? 0.1 : 0.4;
	for(int i = 0; i < 40; i++) asientos[i] = double(rand())/RAND_MAX < probabilidad;

}

class Train {
	Carriage * carriages;
	int numPri;
	int numSeg;
	char * salida;
	char * destino;
	int precioPrimera;
	int precioSegunda;
public:	//lista inicializadores
	Train(int nP, int nS, char * sal, char * dest, int pP, int pS):
	numPri(nP), numSeg(nS), salida(sal), destino(dest), precioPrimera(pP), precioSegunda(pS) {
		carriages = new Carriage[nP + nS];
		for(int i = 0; i < nP + nS; i++) carriages[i].setPrimera(i < nP);
	};
	~Train() { delete [] carriages; }
	Carriage getCarriage(int i) { return carriages[i]; }
	void llenar();
	int totalVentas();
};

int Train::totalVentas() {
	int acum = 0;
	for(int i = 0; i < numPri + numSeg; i++)
		if (carriages[i].getPrimera())
			for(int j = 0; j < 40; j++)
				acum += carriages[i].getAsiento(j) * precioPrimera;
		else for(int j = 0; j < 40; j++)
				acum += carriages[i].getAsiento(j) * precioSegunda;
	return acum;
}

void Train::llenar() {
	for(int i = 0; i < numPri + numSeg; i++)
		carriages[i].ocupar();
}

int main() {
	Train t(4, 7, "Rosario", "Tucuman", 20, 10);
	t.llenar();
	cout << endl << endl << t.totalVentas() << endl;
}
```

Problema 4

Escriba una clase que represente un paquete de información enviado a través de una red. Los paquetes tienen las siguientes características que han sido simplificadas para esta aplicación:

- La dirección IP del emisor.

- La dirección IP del receptor.
- Las direcciones IP son arreglos de 4 chars.
- Un número identificador (ID) del paquete que puede estar entre 0 y 65535.
- El dato enviado representado por un array de bytes. Para este problema se supone que cada paquete tiene una longitud fija de 1000 bytes. Si la información a enviar ocupa menos espacio se rellena con bytes que van alternando ente valores iguales a -1 y 0.

Escriba la clase que represente a uno de estos paquetes. Dicha clase deberá proveer la siguiente funcionalidad:

- Defina un constructor con parámetros para las direcciones de IP del emisor y receptor, el ID del paquete, y un array de tipo byte.
- Implemente el método de acceso para la dirección IP del receptor.
- Escriba un método de modificación para especificar un elemento en el array de datos.
- Sobrecargue el operador '==='. Dos paquetes se consideran iguales si las direcciones IP del destinatario y receptor coinciden al igual que los ID de los paquetes comparados.

Problema 5

Escriba el código de una clase `Candado` que modele un candado con combinación numérica (como los usados para equipaje de viaje). Al utilizar por primera vez dicho candado, puede programársele un número de 3 dígitos del 0 al 9 (deberá almacenarlos en un array) que, será la combinación de seguridad.
La clase `Candado` deberá almacenar también información (en forma de array) del estado actual de los tres dígitos de la combinación (que en un dado instante podrá o no coincidir con los 3 dígitos programados).
Agregue un constructor que inicialice ambos arrays.
Deberá añadir además los siguientes métodos:

- Un método que permita alterar alguno de los 3 dígitos (indicar posición) de la combinación actual.
- Un método, `puedeAbrir`, que retornará una variable booleana de valor `true` en caso que la cerradura pueda abrirse y, `false` en caso contrario.
- Un método `mismaCombinacionActual` que retornará una variable booleana de valor `true` en el caso que la combinación actual del objeto con el cual se invoca este método coincida con la combinación actual de otro `Candado` (deberá pasarlo como argumento al invocarlo).

Deberá escribir una función `main` donde creará dos objetos de tipo `Candado`, c1 y c2. Deberá cambiar uno de los tres dígitos de la combinación actual de c1, mostrar por pantalla: un mensaje que indique si con la combinación actual de c2 que programó se puede abrir o no y otro indicando si c1 y c2 tienen (o no) programados las mismas combinaciones actuales.

Problema 6

Un polinomio es una expresión matemática en forma de suma de potencias de una o más variables multiplicadas por coeficientes. Un polinomio en una variable con coeficientes constantes está dado por:

$$a_n x^n + a_{n-1} x^{n-1} + .. + a_2 x^2 + a_1 x + a_0$$

El grado u orden del polinomio está dado por la potencia más elevada del mismo.
Escriba el código de una clase `Polinomio` que modele un polinomio monovariable. Dicha clase deberá almacenar información sobre el orden del polinomio (mayor o igual a 0) y sobre los coeficientes (usar array de flotantes de doble precisión con de n+1 elementos para el caso de polinomio de orden n) de cada potencia.
Añada a la clase:

- un constructor por defecto
- un constructor de copia
- un constructor donde se le pase como información un array con los coeficientes del polinomio (de 0 a n).

Deberá escribir 2 métodos de acceso que permitan: uno de ellos, leer el orden del polinomio y el otro, leer el coeficiente que corresponda a una determinada potencia, información que deberá pasar como argumento al invocar el método.

Defina además un método, `evaluar`, que evalúe el polinomio en un valor dado.

Sobrecargue los operadores necesarios para poder ejecutar una funciòn `main` como la siguiente:

```
//PruebaPolinomio.cpp
#include "Polinomio.h"
using std::cout;
using std::endl;

int main(void)
{
	double v[]={-2, 3.3, -1.5, 3};
	Polinomio p(3, v);
	cout << endl << "p(x) = " << p << endl;

	double v1[] = {2.0, -1.3, 1.5, -3};
	Polinomio p1(3, v1);
	cout << endl << "p1(x) = " << p1 << endl;

	Polinomio p2(p + p1); // usa sobrecarga de + y constructor de copia
	cout << endl << "p2(p + p1) = " << p2 << endl; // usa sobrecarga de <<

	p2 = p1; // usa sobrecarga de =

	cout << endl << "p(x) += p1(x) = " << (p += p1) << endl; // usa sobrecarga de << y
de +=

	Polinomio p3(-p); // usa constructor de copia y sobrecarga de - unario
	cout << endl << "p3(-p) = " << p3 << endl;

	// sobrecarga de < (un polinomio p es menor que otro polinomio p1 si p evaluado en
0 da un valor menor que p1 evaluado en 0)
	cout << endl << "p(0) < p1(0) = " << (p < p1) << endl;

	return 0;
}
```

Solución:

```
// Polinomio.h
#include <iostream>
using namespace std;
```

```
class Polinomio
{
public:
	Polinomio();
	Polinomio(const Polinomio& p);
	Polinomio(int n, double []);
	int getOrden()const;
	double getCoef(int p)const;
	double* calcular(int limite);
	double evaluar (double) const;
	~Polinomio(void);

	Polinomio& operator= (const Polinomio& p);
	Polinomio operator+ (const Polinomio&) const;
	Polinomio& operator+= (const Polinomio& p);


	bool operator< (const Polinomio& p) const;

	bool operator== (const Polinomio& p);

	friend ostream& operator<< (ostream& o, const Polinomio& p);
	friend Polinomio operator- (const Polinomio& p);


private:
	int orden;
	double *coef;
};


// Polinomio.cpp
#include "Polinomio.h"
#include <math.h>

Polinomio::Polinomio(int n, double *v)
{
	orden=n;
	coef=new double[n+1];
	for(int i=0;i<(n+1);i++)
		coef[i]=v[i];
}

Polinomio::Polinomio(){ coef = 0;}

Polinomio::Polinomio(const Polinomio& p)
{
	orden = p.orden;
	coef = new double[orden+1];
	for(int i=0;i<orden+1;i++)
		coef[i]=p.coef[i];
}
int Polinomio::getOrden()const
{
	return orden;
}
double Polinomio::getCoef(int n) const
{
	return coef[n];
}

double Polinomio::evaluar(double d) const{
```

```
	double res = 0;
	for(int i = 0; i < orden + 1; i++)
		res += pow(d, i) * coef[i];
	return res;
}

bool Polinomio::operator< (const Polinomio& p) const {
	return this->evaluar(0.0) < p.evaluar(0.0);
}

double* Polinomio::calcular(int limite)//no puedo retornar array
{
	double *p=new double[limite+1];
	double suma;
	for(int i=0;i<=limite;i++){
		suma=0;
		for(int j=0;j<=orden;j++)
			suma+=pow((double)i,j)*coef[j];
		p[i]=suma;
	}
	return p;
}
Polinomio::~Polinomio(void)
{
	delete[] coef;
}
Polinomio& Polinomio::operator= (const Polinomio& p)
{
	if(this == &p) return *this;

	delete[] coef;
	coef = new double[p.orden + 1];

	for(int i=0;i<p.orden+1;i++)
		this->coef[i]=p.coef[i];

	this->orden = p.orden;

	return *this;
}


Polinomio& Polinomio::operator+= (const Polinomio& p)
{
	*this = *this + p;
	return *this;
}

bool Polinomio::operator== (const Polinomio& p)
{
	if(this->orden != p.orden) return false;
	int i = 0;
	while((coef[i] == p.coef[i]) && (i < orden + 1)) i++;
	return (i == orden+1);
}


Polinomio Polinomio::operator+ (const Polinomio& p) const {
	int sum_ord = this->orden > p.orden ? this->orden : p.orden;
	int ordenMin = this->orden < p.orden ? this->orden : p.orden;
	double * sum_coef = new double [sum_ord + 1];
	for (int i = 0; i < ordenMin + 1; i++) sum_coef[i] = this->coef[i] + p.coef[i];
```

```
	for (int i = ordenMin + 1; i < sum_ord + 1; i++)
		sum_coef[i] = this->orden > p.orden ? this->coef[i] : p.coef[i];
	int i = sum_ord;
	while(abs(sum_coef[i--]) < 1e-5) sum_ord--;
	return Polinomio(sum_ord, sum_coef);
}


Polinomio operator- (const Polinomio& p) {
	double * c = new double(p.getOrden() + 1);
	for (int i = 0; i < p.getOrden() + 1; i++)
		c[i] = -p.getCoef(i);
	return Polinomio(p.getOrden(), c);
}


ostream& operator<< (ostream& o, const Polinomio& p) {
	for (int i = p.orden; i > 0; i--)
		o << p.coef[i] << "x^" << i << " + ";
	o << p.coef[0];
	return o;
}
```

Problema 7

Se llama “cuadrado mágico” a una matriz cuadrada cuyos elementos tienen la propiedad de estar dispuestos de forma tal que, al sumar cada una de sus filas, cada una de sus columnas y cada una de las dos diagonales se obtiene el mismo resultado.
Un ejemplo de cuadrado mágico es la matriz que se encuentra en cada una de las fachadas de la Sagrada Familia en Barcelona:

1	14	14	4
11	7	6	9
8	10	10	5
13	2	3	15

En este caso la suma de **todas** sus filas, columnas y dos diagonales es 33, la edad a la que murió Jesucristo.

a.- Escriba el código de una clase `Magica` que represente matrices de cualquier cantidad de filas y columnas y se verifique si son o no cuadradas y si son o no mágicas. Deberá usar campos privados.

Dicha clase debe tener un constructor que reciba como argumentos: una matriz de enteros y dos argumentos que indiquen número de filas y número de columnas de la matriz que se pasa como primer argumento. Suponga que las matrices pasadas como primer argumento tienen todas sus filas con la misma cantidad de columnas.

Deberá añadir además los siguientes métodos:

- Un método que retorne una variable cuyo valor indique si la matriz representada por esta clase, es cuadrada o no (es decir, si el número de filas es igual al de columnas, o no).

- Un método que retorne una variable cuyo valor indique la suma de una fila i de la matriz representada por esta clase. Suponga que se pasa como argumento un valor correcto.
- Un método que retorne una variable cuyo valor indique la suma de columna fila i de la matriz representada por esta clase. Suponga que se pasa como argumento un valor correcto.
- Un método que retorne una variable cuyo valor indique la suma de la diagonal principal (es decir, los elementos a_{ii}) de la matriz representada por esta clase. Suponga que este método se va a invocar para una matriz cuadrada.
- Un método que retorne una variable cuyo valor indique la suma de la diagonal secundaria de la matriz representada por esta clase (es decir, los elementos $a_{i(n-1-i)}$, con i=0 hasta (n-1), siendo n el número de filas o columnas de la matriz). Suponga que este método se va a invocar para una matriz cuadrada.
- Un método que retorne una variable cuyo valor indique si la matriz representada por esta clase es un cuadrado mágico o no. Suponga que este método se va a invocar para una matriz cuadrada. Nota: Se recomienda sumar cualquiera de sus filas, columnas o diagonales y luego comparar este valor con las sumas del resto de las filas, columnas y diagonales. Si se encuentra cualquier fila/columna o diagonal cuya suma tenga un valor distinto al de comparación, se puede decir que la matriz no es cuadrado mágico.

b.- Escriba el código de una función main donde creará un objeto de tipo **`Magica`**. Deberá invocar el método que determine si la matriz representada por dicha clase es cuadrada o no y, en caso de serlo, deberá invocar al método que determina si es un cuadrado mágico o no. En caso de serlo deberá mostrar por pantalla un mensaje indicando el valor de la suma de sus filas/columnas/diagonales. En caso contrario deberá mostrar un mensaje por pantalla indicando que no es cuadrado mágico.

Solución:

```
//Magica.h
#pragma once
class Magica
{
public:
	Magica(int **m, int f, int c);
	bool esCuadrada()const;//true es cuadrada, false no es cuadrada
	int sumafilai(int i)const;
	int sumacolumnai(int i)const;
	int sumadiagpcipal()const;
	int sumadiagsec()const;
	bool esMagica()const;
	~Magica(void);
private:
	int** matriz;
	int filas;
	int columnas;
};
//Magica.cpp
#include "Magica.h"


Magica::Magica(int **m, int f, int c)
{
	filas=f;
	columnas=c;
	//reservo espacio para el vector que apunta a las filas
	matriz=new int*[f];
	//reservo memoria para las columnas de cada fila
	for(int i=0;i<c;i++)
```

```
            matriz[i]=new int[c];
      //llena con datos
      for(int i=0;i<filas;i++)
            for(int j=0;j<columnas;j++)
                  matriz[i][j]=m[i][j];
}
bool Magica::esCuadrada()const
{
 if(filas==columnas)return true;
 else return false;
}
int Magica::sumafilai(int i)const
{   int suma=0;
      for(int j=0;j<columnas;j++)
            suma+=matriz[i][j];
      return suma;
}
int Magica::sumacolumnai(int i)const
{
      int suma=0;
      for(int j=0;j<filas;j++)
            suma+=matriz[j][i];
      return suma;
}
int Magica::sumadiagpcipal()const
{
      int suma=0;
      for(int i=0;i<filas;i++)
            suma+=matriz[i][i];
     return suma;
}

int Magica::sumadiagsec()const
{
      int suma=0;
      for(int i=0;i<filas;i++)
            suma+=matriz[0][filas-1-i];
      return suma;
}
bool Magica::esMagica()const
{
      int suma=0;
      for(int i=0;i<columnas;i++)
            suma+=matriz[0][i];
    int i;
      for(i=1;i<filas;i++)
            if(suma!=sumafilai(i))return false;
      for(i=0;i<columnas;i++)
            if(suma!=sumacolumnai(i))return false;
      if(suma!=sumadiagpcipal())return false;
      if(suma!=sumadiagsec())return false;
      return true;
}
Magica::~Magica(void)
{
      //libera la memoria de la matriz
      for(int i=0;i<filas;i++)
      delete[]matriz[i];//libera columnas
      delete[]matriz;//libera filas
}
//pruebaMagica.cpp
#include <iostream>
```

```
#include "Magica.h"
using std::cout;

int main(void)
{
	int matriz[4][4]={{1,14,14,4},{11,7,6,9},{8,10,10,5},{13,2,3,15}};
	int **p;
	p=new int*[4];
	for(int i=0;i<4;i++)
		p[i]=new int[4];
	for(int i=0;i<4;i++)
		for(int j=0;j<4;j++)
		p[i][j]=matriz[i][j];
	Magica c(p, 4,4);
	if(c.esCuadrada())cout<<"La matriz es cuadrada\n";
	else cout<<"La matriz no es cuadrada\n";
	if(c.esCuadrada()){
		if(c.esMagica())
			cout<<"La suma de cualquier fila, columna o diagonal es:
"<<c.sumadiagpcipal()<<"\n";
		else
			cout<<"La matriz no es magica\n";
	}
	return 0;
}
```

Problema 8

Un anillo (representado por la clase `Anillo`) es un par de círculos (representados por la clase `Circulo`) concéntricos especificado por un centro (existe una clase `Punto` para representar cualquier punto del plano) el radio del círculo interior y el ancho del anillo (diferencia entre el radio del círculo exterior y el radio del círculo interior). Dado el código del archivo de cabecera de las clases `Anillo`, `Punto` y `Circulo` deberá programar los archivos `cpp` asociados. El área del anillo es la diferencia entre el área del círculo externo y el área del círculo interno. Para mover un anillo, se deben mover los dos círculos.

```
#pragma once
class Punto{
public:
	Punto(double cx=0.0, double cy=0.0);
	~Punto(void);
	double getX(void);
	double getY(void);
	void setX(double cx);
	void setY(double cy);
	void mover(double dx, double dy);
	void mostrar(void);
private:
	double x;
	double y;};


#pragma once
#include "Punto.h"
class Circulo
{
public:
	Circulo(double r=1.0, Punto p = Punto(0.0, 0.0));
	double getRadio(void);
```

```
        void setRadio(double r);
        double area(void);
        void mostrar(void);
        void mover (Punto nuevoCentro);
        ~Circulo(void);
private:
        double radio;
        Punto centro;
};

#pragma once
#include "Circulo.h"
class Anillo
{
public:
        Anillo(Punto p = Punto(0.0, 0.0), double radio=1.0, double a=1.0);
        double getAncho(void);
        void setAncho(double a);
        double area(void);
        void mover(Punto p);
        void mostrar(void);
        ~Anillo(void);
private:
        Circulo interior;
        Circulo exterior;
        double ancho;
};
```

Problema 9

Defina una clase para representar una pila de enteros. Una pila es una lista de items que permite agregar (push) o sacar (pop) sólo de un extremo, operando con el principio LIFO (last-in, first out). Por ejemplo, si la pila contiene [10 4 16 20], pop() debe retornar 10, y luego la pila queda [4 16 20]. Un subsecuente push(13) debe dejar la pila en [13 4 16 20]. No se puede obtener un item que no esté en el top, salvo que se quiten todos los items que están sobre él. La clase debe implementar las funciones push(), pop() y una función print() para mostrar el estado de la pila. Guarde la lista internamente como un arreglo. Escriba un programa de test para verificar el correcto funcionamiento de la clase.

Problema 10

Modifique la clase pila del ejercicio 13 de forma que el tamaño de la misma sea especificado mediante un constructor con reserva de memoria. Agregue lo que sea necesario. Pruebe el funcionamiento.

Solución:

```
#include <iostream>
using std::cout;
using std::endl;

class CStack
{
public:
   CStack(int n = 10);
   ~CStack();
   void push(int i);
   int pop();
   void print();
```

```
private:
   int* pList;
   int size;
   int next;
};

CStack::CStack(int n) : next(0), size(n)
{
   pList = new int[size];
}

CStack::~CStack()
{
   delete [] pList;
}

void CStack::push(int i)
{
   if (next < 99)
      pList[next++] = i;
}

int CStack::pop()
{
   return pList[--next];
}

void CStack::print()
{
   cout << '[';
   for(int i=next-1 ; i>=0 ; i--)
      cout << ' '<< pList[i];
   cout << " ]\n";
}

int main()
{
   CStack s(20);

   s.print();

   s.push(5);
   s.push(10);
   s.push(8);

   s.print();

   cout << "top of stack=" << s.pop() << endl;

   s.print();

   return 0;
}
```

Problema 11

Cree una clase `Avión` que almacene el nombre del vuelo que realiza y el destino (añadir métodos de acceso y modificación de dichos datos miembros) y permita mostrar por pantalla los datos asociados al mismo. Luego cree dos clases derivadas: `AvionPasajeros` y `AvionHidrante`, con los miembros necesarios para que el siguiente programa funcione correctamente:

```
#include "AvionHidrante.h"
#include "AvionPasajeros.h"
#include <iostream>
using std::cout;

int main(void)
{
	AvionPasajeros ap("AZ19B2", "Londres", 2400, 270.5);
	AvionHidrante ah("BT3533P8", "Bariloche", 500);
	ap.mostrarDatos();
	ah.mostrarDatos();
	ah.descargarAgua(300);
	ah.mostrarDatos();
	ah.descargarAgua();
	ah.mostrarDatos();
	return 0;
}
```

De los aviones de pasajeros (además de los datos miembros del avión) debe almacenar información sobre la cantidad de kilos de equipaje que transporta y la duración prevista del vuelo (en minutos). De los aviones hidrantes interesa almacenar la cantidad de agua que transportan (en litros) y la cantidad de litros de agua actuales. La función `descargarAgua ()` debe permitir disminuir la cantidad de agua que se le pase como argumento, sino se le pasa argumento deberá soltar todo el agua restante.

Solución:

```
//Avion.h
#pragma once
class Avion
{
public:
	Avion(char* nv, char* d);
	void mostrarDatos(void);
	char* getNombreVuelo(void);
	char* getDestino(void);
	void setDestino(char* d);
	void setNombreVuelo(char* nv);
	~Avion(void);
protected:
	char* nombrevuelo;
	char* destino;
};
//Avion.cpp
#include "Avion.h"
#include <string.h>
#include <iostream>
using std::cout;

Avion::Avion(char* nv, char* d)
{
nombrevuelo=new char[strlen(nv)+1];
strcpy_s(nombrevuelo, strlen(nv)+1,nv);
destino=new char[strlen(d)+1];
strcpy_s(destino, strlen(d)+1,d);
}
char* Avion::getDestino(void){
	return destino;
}
```

```
char* Avion::getNombreVuelo(void)
{
	return nombrevuelo;
}
void Avion::setDestino(char* d)
{
	delete [] destino;
	destino=new char[strlen(d)+1];
	strcpy_s(destino, strlen(d)+1,d);
}
void Avion::setNombreVuelo(char* nv)
{
	delete [] nombrevuelo;
	nombrevuelo=new char[strlen(nv)+1];
	strcpy_s(nombrevuelo, strlen(nv)+1,nv);
}
void Avion::mostrarDatos(void)
{
cout<<"El vuelo se llama: "<<nombrevuelo<<" y tiene destino: "<<destino<<"\n";
}
Avion::~Avion(void)
{

}
//AvionHidrante.h
#pragma once
#include "avion.h"
class AvionHidrante :
	public Avion
{
public:
	AvionHidrante(char* nv, char* d, int l);
	void descargarAgua(int l);
	void descargarAgua();
	void mostrarDatos(void);
	~AvionHidrante(void);
private:
	int litrosAgua;
	int capacidad;
};
//AvionHidrante.cpp
#include "AvionHidrante.h"
#include <iostream>
using std::cout;

AvionHidrante::AvionHidrante(char* nv, char* d, int c):Avion(nv, d)
{
	capacidad=c;
	litrosAgua=c;
}


void AvionHidrante::descargarAgua(int l){
	litrosAgua -=l;
}
void AvionHidrante::descargarAgua()
{
	litrosAgua=0;
}
```

```
void AvionHidrante::mostrarDatos()
{
	Avion::mostrarDatos();
	cout<<"La capacidad en litros de agua es: "<<capacidad
		<<". Actualmente le quedan: "<<litrosAgua<<"litros de agua\n";
}
AvionHidrante::~AvionHidrante(void)
{
}
//AvionPasajeros
#pragma once
#include "avion.h"
class AvionPasajeros :
	public Avion
{
public:
	AvionPasajeros(char* nv, char* d, int du, double p);
	void mostrarDatos(void);
	~AvionPasajeros(void);
private:
	int duracionVuelo; //en minutos
	double pesoEquipaje;
};
//AvionPasajeros.cpp
#include "AvionPasajeros.h"
#include <iostream>
using std::cout;

AvionPasajeros::AvionPasajeros(char* nv, char* d, int du, double p):Avion(nv,d)
{
	duracionVuelo=du;
	pesoEquipaje=p;
}

void AvionPasajeros::mostrarDatos(void)
{
	Avion::mostrarDatos();//reutilizo código
	cout<<"La duracion del mismo: "<<duracionVuelo<<" minutos, con capacidad maxima para
equipaje: "<<
		pesoEquipaje<<"\n";
}

AvionPasajeros::~AvionPasajeros(void)
{
}
//pruebaAvion.cpp
#include "AvionHidrante.h"
#include "AvionPasajeros.h"
#include <iostream>
using std::cout;

int main(void)
{
	AvionPasajeros ap("AZ19B2", "Londres", 2400, 270.5);
	AvionHidrante ah("BT3533P8", "Bariloche", 500);
	ap.mostrarDatos();
	ah.mostrarDatos();
	ah.descargarAgua(300);
	ah.mostrarDatos();
	ah.descargarAgua();
```

```
	ah.mostrarDatos();
	return 0;
}
```

Problema 12

Dado el archivo de cabecera de la clase abstracta `Compuerta`:

```
#pragma once
class Compuerta
{
public:
	Compuerta(int nro);
	int getNroentradas(void);//retorna nro.entradas
	int getEntradaNro(int nro);//retorna la entrada nro.
	void setEntradaNro(int nro, int val);//cambia la entrada nro con val
	virtual int eval(void)=0;//evalua la salida de la compuerta
	static const int FALSE=0;
	static const int TRUE=1;
	static const int UNKNOWN=2;
	~Compuerta(void);
protected:
	int nroentradas;//número de entradas de la compuerta
	int* entradas;//vector con los valores presentes en todas las entradas de la compuerta
};
```

complete el archivo `cpp` de dicha clase. Escriba el código de la clase concreta `CompuertaAND` que represente al tipo de compuertas que realizan la función lógica `AND` de todas sus entradas. Implemente la aplicación `Prueba` con una función `main` donde se cree una instancia de `CompuertaAND` (usando un puntero de tipo `Compuerta`) para evaluar el resultado.
Nota: si cualquier entrada de esta compuerta tiene un valor `UNKNOWN`, el producto lógico que realiza la compuerta tendrá este valor.

Problema 13

Dado el archivo de cabecera de la clase Volumen:

```
#pragma once
class Volumen
{
public:
	Volumen(char* t="", char* i="", int n=1);
	Volumen (const Volumen& v);
	char* getTitulo(void);
	char* getIsbn(void);
	int getNroEjemplares(void);
	virtual void mostrarDatos(void);
	void setDatos(char* t, char*i, int n);//seteo de los datos del libro
	~Volumen(void);
protected:
	char* titulo;
	char* isbn;
	int nroejemplares;//cant.de ejemplares de cada volumen
};
```

Cree dos clases que deriven de `Volumen`: `Libro` y `Revista`. De la clase `Libro` (además de `título` e `ISBN`) interesa almacenar el `autor` y `editorial` del mismo. De la clase `Revista` interesa (además de `título` e `ISBN`) almacenar el `año de publicación y número` de la misma.

Además deberá crear una clase `Biblioteca` como un array de punteros a `Volumen` de un máximo a determinar por el alumno (capacidad de almacenamiento de la biblioteca) añada una función miembro para mostrar todos los volúmenes (libros y revistas) que tenga la biblioteca y otra `bool añadirVolumen(Volumen* v)` que añada el volúmen a que apunta `v (Libro o Revista)` y retorne `true` si pudo añadir el volúmen (aún no se alcanzó la capacidad máxima de la biblioteca) y `false` en caso contrario.
Cree una aplicación `Prueba` con una función `main()` que cree algunos libros y revistas (como punteros a `Volumen`) y los añada en una `Biblioteca` previamente creada. Muestre los datos de los volúmenes disponibles en dicha `Biblioteca`.

Solución:

//Volumen.h

```
#pragma once
class Volumen
{
public:
	Volumen(char* t="", char* i="", int n=1);
	Volumen (const Volumen& v);
	char* getTitulo(void);
	char* getIsbn(void);
	int getNroEjemplares(void);
	virtual void mostrarDatos(void);
	void setDatos(char* t, char*i, int n);
	~Volumen(void);
protected:
	char* titulo;
	char* isbn;
	int nroejemplares;
};
```

//Volumen.cpp

```
#include "Volumen.h"
#include <iostream>
using std::cout;
Volumen::Volumen(char* t, char* i, int n)//ojo ocurre efecto division en porciones (12.2.3)
{
	titulo=new char[strlen(t)+1];
	strcpy_s(titulo, strlen(t)+1,t);
	isbn=new char[strlen(i)+1];
	strcpy_s(isbn, strlen(i)+1,i);
	nroejemplares=n;
}
Volumen::Volumen(const Volumen& v)
{
	titulo=new char[strlen(v.titulo)+1];
	strcpy_s(titulo, strlen(v.titulo)+1,v.titulo);
	isbn=new char[strlen(v.isbn)+1];
	strcpy_s(isbn, strlen(v.isbn)+1,v.isbn);
	nroejemplares=v.nroejemplares;
	cout<<"Se llama al constructor por copia de Volumen";
}
char* Volumen::getTitulo(void)
{
	return titulo;
}
char* Volumen::getIsbn(void)
{
	return isbn;
}
void Volumen::mostrarDatos(void)
```

```
{
	cout<<"El título del volumen es: "<<titulo<< " y el ISBN: "<<isbn
		<<"\n";
}
int Volumen::getNroEjemplares(void)
{
	return nroejemplares;
}
void Volumen::setDatos(char* t, char* i, int n)
{
	titulo=new char[strlen(t)+1];
	strcpy_s(titulo, strlen(t)+1,t);
	isbn=new char[strlen(i)+1];
	strcpy_s(isbn, strlen(i)+1,i);
	nroejemplares=n;
}
Volumen::~Volumen(void)
{
}
```

//Revista.h

```
#pragma once
#include "volumen.h"
class Revista :
	public Volumen
{
public:
	Revista(char* t="", char* i="", int a=0, int nror=0, int n=1);
	int getaño(void);
	int getnrorevista(void);
	virtual void mostrarDatos(void);
	void setDatos(char* t, char* i, int a, int nror, int n);
	~Revista(void);
private:
	int año;
	int nrorevista;
	};
```

//Revista.cpp

```
#include "Revista.h"
#include <iostream>
using std::cout;

Revista::Revista(char* t, char* i, int a, int nror, int n):Volumen(t,i,n)
{
	año=a;
	nrorevista=nror;
}
int Revista::getaño(void)
{
	return año;
}
int Revista::getnrorevista(void)
{
	return nrorevista;
}
void Revista::mostrarDatos(void)
{
	Volumen::mostrarDatos();
	cout<<"El año de la revista es: "<<año<< ", el numero es: "<<nrorevista
		<<" y la cantidad de ejemplares que hay es: "<<nroejemplares
		<<"\n";
}
```

```
void Revista::setDatos(char* t, char* i, int a, int nror, int n)
{
	Volumen::setDatos(t,i,n);
	año=a;
	nrorevista=nror;
}
Revista::~Revista(void)
{
}
```

//Libro.h

```
#pragma once
#include "volumen.h"
class Libro :
	public Volumen
{
public:

	Libro(char* t="", char* i="", char* a="", char* e="",int n=1);
	Libro(const Libro& otro);
	char* getautor();
	char* geteditorial();
	virtual void mostrarDatos(void);
	void setDatos(char* t, char* i, char* a, char* e,int n);
	~Libro(void);
private:
	char* autor;
	char* editorial;
};
```

//Libro.cpp

```
#include "Libro.h"
#include <iostream>
using std::cout;

Libro::Libro(char* t, char* i, char* a, char* e,int n):Volumen(t,i,n)
{
	autor=new char[strlen(a)+1];
	strcpy_s(autor, strlen(a)+1,a);
	editorial=new char[strlen(e)+1];
	strcpy_s(editorial, strlen(e)+1,e);
	}
Libro::Libro(const Libro& l):Volumen(l)
{
	autor=new char[strlen(l.autor)+1];
	strcpy_s(autor, strlen(l.autor)+1,l.autor);
	editorial=new char[strlen(l.editorial)+1];
	strcpy_s(editorial, strlen(l.editorial)+1,l.editorial);
	cout<<"Se llama al constructor por copia de Libro";
}
char* Libro::getautor(void)
{
	return autor;
}
char* Libro::geteditorial(void)
{
	return editorial;
}
void Libro::mostrarDatos(void)
{
	Volumen::mostrarDatos();
	cout<<"El autor del libro es: "<<autor<< ", la editorial: "<<editorial<<" y la cantidad de
volúmenes: "
		<<nroejemplares
```

```
		<<"\n";
}
void Libro::setDatos(char* t, char* i, char* a, char* e,int n)
{
	Volumen::setDatos(t,i,n);
	autor=new char[strlen(a)+1];
	strcpy_s(autor, strlen(a)+1,a);
	editorial=new char[strlen(e)+1];
	strcpy_s(editorial, strlen(e)+1,e);
}

Libro::~Libro(void) { }
```

//Biblioteca.h

```
#pragma once
#include "volumen.h"
class Biblioteca
{
public:
	Biblioteca(void);
	void mostrarDatos(void);
	bool añadirVolumen(Volumen* v);
	~Biblioteca(void);
private:
	static const int MAX=100;
	Volumen* volumenes[MAX];
	int i;
};
```

//Biblioteca.cpp

```
#include "Biblioteca.h"


Biblioteca::Biblioteca(void)
{
	i=0;
}

void Biblioteca::mostrarDatos(void)
{
	for(int j=0;j<i;j++)
		volumenes[j]->mostrarDatos();
}
bool Biblioteca::añadirVolumen(Volumen* v){
	bool agregado=false;
	if(i<MAX){
		volumenes[i]=v;
		i++;//queda apuntando al siguiente
		agregado=true;
	}
	return agregado;
}

Biblioteca::~Biblioteca(void)
{
}
```

//Biblioteca.cpp

```
#include <iostream>
#include "Biblioteca.h"
#include "Revista.h"
#include "Libro.h"
using std::cout;
using std::endl;
```

```
int main()
{
	Libro l("Programacion en lenguajes estructurados", "970-15-1165-4","Ma.Asuncion Criado
Clavero", "Alfaomega", 1);
	Revista r("Lugares", "9788435061599", 2011,189,2);
	Biblioteca b;
	b.añadirVolumen(&l);
	b.añadirVolumen(&r);
	b.mostrarDatos();
	return 0;
}
```

Problema 14

Un tren de carga lleva dos clases de vagones: `VagonCaja` que tienen forma de prisma rectangular y `VagonTanque` que tienen forma cilíndrica.
Deberá diseñar tres clases para representar vagones de tren, una clase base (`Vagon`) abstracta con un dato miembro que represente la `longitud` de los vagones y dos clases derivadas (`VagonCaja` y `VagonTanque`) para representar los dos tipos de vagones. Para el vagón de tipo `VagonCaja` deberá almacenar su `alto, ancho` (aparte de la `longitud`). Para el vagón de tipo `VagonTanque` deberá almacenar su `radio` (aparte de su `longitud`). Las tres clases deben poseer un método `volumen()` que retorna un valor de tipo `double` que representa la capacidad en metros cúbicos del vagón. Parte de su trabajo será decidir si el método `volumen()` en cada una de las clases es declarado abstracto o no.
Recuerde que el volumen de un prisma rectangular es: `alto * ancho * longitud` y el del cilindro es `PI * r`^{2} `* longitud`.
Cree una clase `Tren` que represente a un tren de carga con sus vagones. Utilice un array de punteros de tipo `Vagon`. La clase debe tener una función miembro `bool añadirVagon(Vagon* v);` que permita añadir un vagón de algún tipo al `Tren` (si es que no supera la cantidad de vagones límite del tren) y otra que retorne el volumen total del tren.
Cree una aplicación de prueba que permita crear trenes y calcular la capacidad total de carga de cada tren.

Problema 15

Dada la siguiente clase abstracta, cuyo archivo de cabecera es el siguiente:

```
#pragma once
class Persona
{
public:
	Persona(char* n, double e, double p);
	char* getNombre(void);
	double getEstatura(void);
	double getPeso(void);
	double IMC(void);
	virtual double pesoteorico(void)=0;
	virtual char* estadoOMS(void)=0;
	virtual void mostrardatos();
	~Persona(void);
protected:
	char* nombre;
	double estatura;
	double peso;
};
```

donde el Indice de Masa Corporal (`IMC`) se calcula dividiendo los kilogramos de peso por el cuadrado de la estatura en metros. Complete el archivo `cpp` de dicha clase y los siguientes archivos de cabecera de las clases derivadas `Hombre` y `Mujer`:

```
#pragma once
#include "persona.h"
class Hombre :      public Persona
{
public:
      Hombre(char* n, double e, double p, bool c);
      //completar
private:
      bool corpulento;
};
#pragma once
#include "persona.h"
class Mujer : public Persona
{
public:
      Mujer(char* n, double e, double p, int nh);
      //completar
private:
      int nrohijos;
};
```

Debe tenerse en cuenta que:

a) El peso teórico se calcula como:

- Mujeres: 50 kg para los primeros 1.5 m de altura más 2.5 kg por cada 2.5 cm adicionales, mas 0.65 Kg por un hijo ó 1.3 Kg por 2 hijos o mas.
- Hombres: 53 kg para los primeros 1.5 m de altura más 3 kg para cada 2.5 cm adicionales, mas 10% si el hombre es corpulento.

b) El Indice de Masa Corporal (IMC) da origen a la siguiente clasificación acorde a la OMS:

- < 18,5: Peso Bajo (Indica delgadez, desnutrición o algún problema de salud)
- hasta 24,9 (hombres) o 23,9 (mujeres). Peso normal (Está catalogado como saludable)
- hasta 29.9 hombres ó 28.9 mujeres: Sobrepeso (u Obesidad leve)
- hasta 40 hombres ó 37 mujeres: Obesidad grado 2.
- > 40 hombres ó > 37 mujeres: Obesidad severa o grado 3.

El código a completar debe ser tal que haga compilar y ejecutar la siguiente aplicación:

```
#include "Hombre.h"
#include "Mujer.h"
#include <iostream>
using std::cout;

int main(void){
      Persona* personas[4];
      personas[0]=new Hombre("Javier",1.75,83.0,false);
      personas[1]=new Mujer("Marisa",1.80,72.0,0);
      personas[2]=new Hombre("Juan",1.70,92.0,false);
      personas[3]=new Hombre("Pedro",1.95,110.0,true);
      for(int i=0;i<4;i++)
            personas[i]->mostrardatos();
      return 0;
}
```