UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDSUTRIAL

Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada

APELLIDOS

NOMBRE Nº Mat.

ASIGNATURA: SISTEMAS INFORMÁTICOS INDUSTRIALES

CURSO 4º GRUPO Diciembre 2014

Calificación

2. Problema de Algoritmia (5 puntos - 15 minutos)

El mapa de celdas que figura a continuación describe las posibles trayectorias de un agente robótico situado en una celda cualquiera. Las celdas con una cruz son inaccesibles para el robot, que se desplaza de celda en celda siguiendo las direcciones de los 4 puntos cardinales (derecha, izquierda, arriba y abajo -nunca en diagonal). El robot parte de la celda 15 y *se quiere recorrer todas las celdas de manera sistemática*.

1	2	3	4
5		7	8
9	10		12
13	14	15	

Se pide:

1. Complete adecuadamente el código que figura a continuación sustituyendo *** por el código adecuado (**3 puntos**)

```
#include "pablodev/courses/breadth_first_search.h"
#include "pablodev/courses/depth_first_search.h"
#include "pablodev/graph/graph.h"
#include <vector>
#include <iterator>

using namespace std;

#define SOURCE 15

int inc_one(int& v){
        return v+1;
}

int main(){
        vector<int> v;

        //codifica el grafo con índices de vértices que empiezan en 0
        ugraph g("cellmap_4x4.txt");
        g.print_data();

        //lanza el algoritmo primero-en-profundidad
        BrFS<ugraph> bf(g,SOURCE-1);
        v=bf.run();                          //retorna la lista de celdas

        //incrementa en una unidad los índices de los vértices
        transform(v.begin(), v.end(), v.begin(), inc_one);

        //muestra en pantalla la lista de vértices
        copy(v.begin(), v.end(), ostream_iterator<int>(cout, " "));
        cout<<endl;
}
```

2. Recorrer sistemáticamente las celdas ¿es un problema de complejidad polinomial o exponencial? Justifique brevemente la respuesta. (**2 puntos**)

SOLUCION: Complejidad polinomial: hay que calcular las celdas vecinas de todas las celdas y como mucho hay un número cuadrático de relaciones de adyacencias respecto al número total de celdas.

3. Escriba la lista de vértices devueltos en *v* tras la instrucción *v=bf.run();* (**2 puntos**)

SOLUCION: 15 14 10 13 9 5 1 2 3 4 7 8 12

4. Grado máximo y mínimo del grafo codificado en *cellmap_4x4.txt*. (**1 punto**)

SOLUCION: Grado máximo 3, Grado mínimo 0 (los obstáculos)

5. Número de clique del grafo. Justifique brevemente la respuesta indicando una posible interpretación del número de clique en este contexto. (**2 puntos**)

SOLUCION: Número de clique: 2 No es posible que exista un triángulo, ya que esto implicaría que se podría alcanzar a partir de cualesquiera de las celdas de ese triángulo las otras dos y la geometría del mapa de celdas no lo permite.

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDSUTRIAL

Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada

APELLIDOS

NOMBRE Nº Mat.

ASIGNATURA: SISTEMAS INFORMÁTICOS INDUSTRIALES

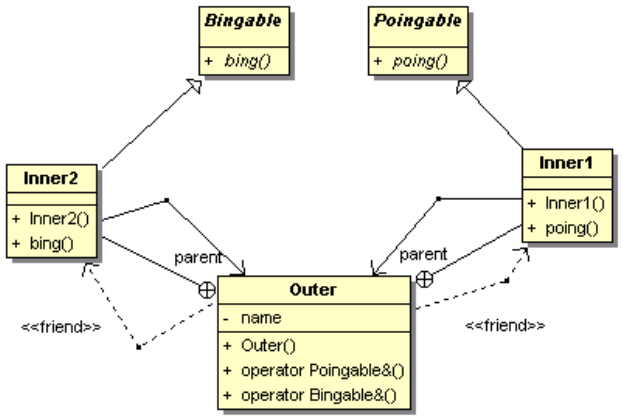
CURSO 4º GRUPO Diciembre 2014

Calificación

3. Problema de Análisis y Diseño Orientado a Objetos (10 puntos - 30 minutos)

Para el código que se adjunta, se pide.

1. Ingeniería inversa (2.5 puntos).
2. Resultado de la ejecución del programa en la consola del sistema (2.5 puntos).



poing called for Ping Pong
bing called for Ping Pong

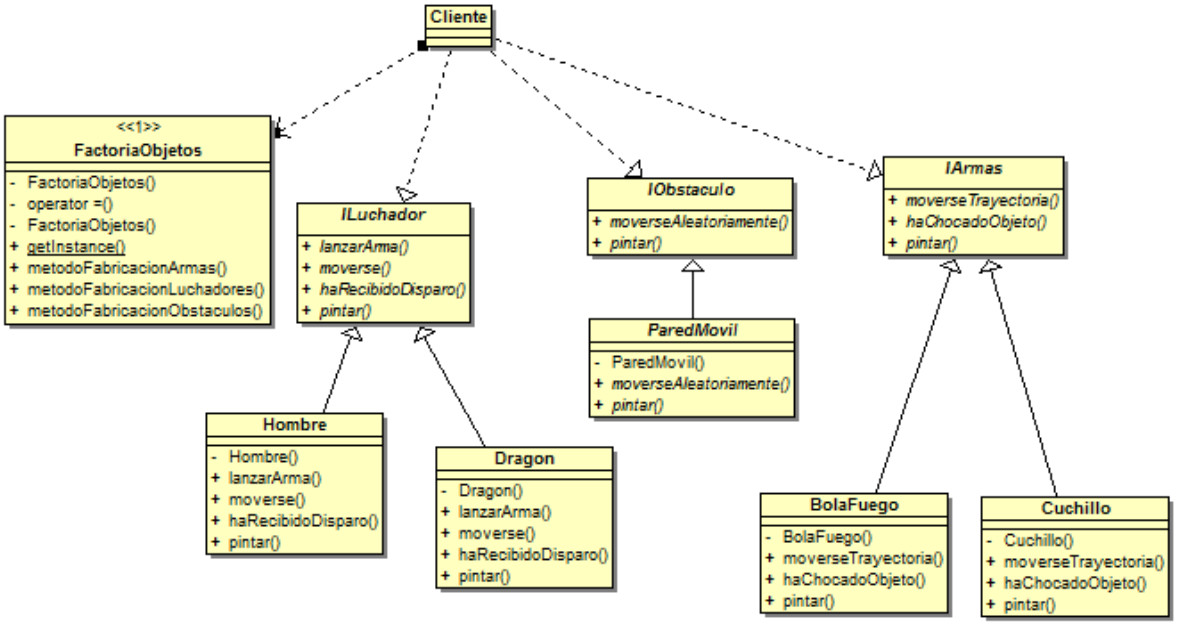
```
#include <iostream>
#include <string>
using namespace std;
class Poingable {
public:
  virtual void poing() = 0;
};
class Bingable {
public:
  virtual void bing() = 0;
};

class Outer {
  string name;
  class Inner1;
  friend class Outer::Inner1;
  class Inner1 : public Poingable {
    Outer* parent;
  public:
    Inner1(Outer* p) : parent(p) {}
    void poing() {
      cout << "poing called for "
        << parent->name << endl;
    }
  } inner1;
  class Inner2;
  friend class Outer::Inner2;
  class Inner2 : public Bingable {
    Outer* parent;
  public:
    Inner2(Outer* p) : parent(p) {}
    void bing() {
      cout << "bing called for "
        << parent->name << endl;
    }
  } inner2;
public:
  Outer(const string& nm)
  : name(nm), inner1(this), inner2(this) {}
  operator Poingable&() { return inner1; }
  operator Bingable&() { return inner2; }
};

void callPoing(Poingable& p) {
  p.poing();
}
void callBing(Bingable& b) {
  b.bing();
}

int main() {
  Outer x("Ping Pong");
  callPoing(x);
  callBing(x);
}
```

Para el DCD adjuntado se pide:



3. Definición en C++ de las clases ILuchador, Hombre,Dragon (2.5 puntos).
4. Implementación en C++ de la FactoriaObjetos (2.5 puntos).

```
#ifndef _LUCHADORES_INC_
#define _LUCHADORES_INC_

class FactoriaObjetos;

typedef enum{HOMBRE,DRAGON}
TipoLuchadores;

class ILuchador
{
public:
 virtual void lanzarArma() = 0;
 virtual void moverse() = 0;
 virtual bool haRecibidoDisparo() = 0;
 virtual void pintar() = 0;
};

class Hombre : public ILuchador
{
 friend class FactoriaObjetos;
 Hombre();
public:
 virtual void lanzarArma();
 virtual void moverse();
 virtual bool haRecibidoDisparo();
 virtual void pintar();
};

class Dragon : public ILuchador
{
 friend class FactoriaObjetos;
 Dragon();
public:
 virtual void lanzarArma();
 virtual void moverse();
 virtual bool haRecibidoDisparo();
 virtual void pintar();
};

#endif
```

```
#ifndef _FACTORIA_INC_
#define _FACTORIA_INC_

#include "Luchadores.h"
#include "Armas.h"
#include "Obstaculos.h"

class FactoriaObjetos {
 FactoriaObjetos(){}; // Para desactivar
 void  operator=(FactoriaObjetos&) {};
 FactoriaObjetos(const FactoriaObjetos&) {};
public:
 static FactoriaObjetos& getInstance() {
 static FactoriaObjetos unicaInstancia;
 return unicaInstancia;
}
IArma* metodoFabricacionArmas (TipoArmas tipo) {
if(tipo == BOLAFUEGO) return new BolaFuego;
      else if(tipo == CUCHILLO) return new Cuchillo;
      else return NULL;}
ILuchador* metodoFabricacionLuchadores (TipoLuchador tipo)
{
      if(tipo == HOMBRE) return new Hombre;
      else if(tipo == DRAGON) return new Dragon;
      else return NULL;}
IObstaculo * metodoFabricacionObstaculos (TipoObstaculos
tipo) {
      if(tipo == PAREDMOVIL) return new ParedMovil;
      else return NULL; }
};
#endif
```

escuela técnica superior de ingeniería y diseño industrial
UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDSUTRIAL
Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada

APELLIDOS

NOMBRE Nº Mat.

ASIGNATURA: SISTEMAS INFORMÁTICOS INDUSTRIALES

CURSO 4º GRUPO Diciembre 2014

Calificación

4. **Problema de Sistemas Operativos (10 puntos - 30 minutos)**

Referente al capítulo de gestión de memoria responda a las siguientes preguntas

- **P1** (1 p) Explique la diferencia de compilar un ejecutable con la opción -static o sin ella.
- **P2** (1 p) Cite las diferentes secciones de un fichero ejecutable, qué contiene cada una de ellas y como se expanden en el mapa memoria al cargarse en un proceso.

Escriba dicho mapa de memoria de un proceso en la siguiente tabla. Se pone a modo de ejemplo la región de código. Escriba en dicha tabla las regiones y sus características:

- **P3** (1 p) Variables Globales con valor inicial
- **P4** (1 p) Variables Globales sin valor inicial
- **P5** (1 p) Heap
- **P6** (1 p) Biblioteca dinámica
- **P7** (1 p) Variables Globales de la biblioteca dinámica
- **P8** (1 p) Pila

Nombre de la región	Permisos de la región	Procedencia de los datos	¿Se rellena de ceros?	¿Es una región dompartida o privada?	Posibles ubicaciones de las páginas de esta región	¿Puede crecer?
Código	R, X	del fichero ejecutable	No	Compartida	Memoria o ejecutable	No
…						

- **P9** (1 p) Para el caso de la pila, dibuje qué contiene el bloque de activación cuando el proceso se encuentra en la función mail.
- **P10** (1 p) Suponiendo que el proceso se encuentra dentro de int funcion(int a, float b), agregue el bloque de activación en el dibujo anterior de la pila.

- P1: Con la opción -static incluye todo el código de las bibliotecas dinámicas en el propio ejecutable. Por tanto éste es mucho más grande y también su imagen en memoria cuando se carga un proceso con dicho ejecutable. Como ventaja es mucho más rápido al no tener que referenciar y cargar código de bibliotecas dinámicas. Si la opción -static se crea un ejecutable normal que utilizará bibliotecas dinámicas.
- P2: Cabecera y secciones. Dentro de esta última está el código, variables con valor inicia y tabla de símbolos. En la cabecera se almacena el número mágico que identifica el formato del ejecutable, el valor de los registros de la CPU en el instante inicial y un mapa de donde se encuentra cada sección dentro del fichero ejecutable. En la sección de código, la más grande, se encuentra el código máquina del programa. El la sección de datos se encuentra aquellas variables globales con valor inicial, así como las constantes (cadenas de

caracteres). Finalmente en la **Tabla de Símbolos** se encuentra información necesaria para el montaje dinámico y depuración.

- P3, P4, P5, P6, P7 Y P8 → cada elemento de la siguiente tabla vale un sexto de punto

Nombre	Permisos	Procedencia de los datos	¿Se rellena de ceros?	compartida o privada?	Posibles ubicaciones	¿Puede crecer?
v.g.c.v.i	R, W	del fichero ejecutable. Resto hasta completar página se rellena de ceros	No	Privada	Swap o Marco	No
v.g.s.v.i	R, W	ceros	Si	Privada	Swap o Marco	No
Heap	R, W	ceros	Si	Privada	Swap o Marco	Si
Biblioteca Dinámica	R, X	Del fichero de la biblioteca dinámica	No	Compartida	Fichero o Marco	No
v.g.v.d.	R, W	Del fichero de la biblioteca dinámica. Resto se rellena de ceros	Léase casilla anterior	Privada	Swap o Marco	No
Pila	R, W	Bloque activación de main(), variables de entorno y lo demás se rellena de ceros	Léase casilla anterior	Privada	Swap o Marco	Si

- P9: Al crearse el proceso la pila tendrá los siguientes valores:

- P10: Al llamar a la función, la pila se incrementará con respecto a la imagen anterior con el tamaño de:
  - Un entero del parámetro a
  - Un float del parámetro b
  - Un entero del valor devuelto por la función
  - La dirección de memoria donde vuelva la ejecución cuando acabe la función.