Junio-2008-Solucion.pdf

fluxneon

Programación Orientada a Objetos

2º Grado en Ingeniería Informática

Escuela Superior de Ingeniería
Universidad de Cádiz

Programación Orientada a Objetos

Examen final

Curso 2007-08 **Martes 10 de junio de 2008**

1. Sean las siguientes clases que permiten representar árboles binarios de búsqueda. **2,5 p**

```
template <class T> class arbolPropio;

template <class T> class arbol {
public:
    arbol();
    arbol(const T& r);
    arbol(const T& r, const arbol& sai, const arbol& sad);
    ~arbol();
    bool vacio() const { return !a; }
    const T& raiz() const {return a->raiz(); }
    const arbol& subarbolIzquierdo() const { return a->subarbolIzquierdo(); }
    const arbol& subarbolDerecho() const { return a->subarbolDerecho(); }
    // ...
    void inserta(const T& e);
private:
    arbolPropio<T>* a;
    arbol& subarbolIzquierdo() { return a->subarbolIzquierdo(); }
    arbol& subarbolDerecho() { return a->subarbolDerecho(); }
};

template <class T> class arbolPropio {
public:
    arbolPropio(const T& r);
    arbolPropio(const T& r, const arbol<T>& sai, const arbol<T>& sad);
    T& raiz() { return r; }
    arbol<T>& subarbolIzquierdo() { return sai; }
    arbol<T>& subarbolDerecho() { return sad; }
private:
    T r;
    arbol<T> sai, sad;
};
```

a) Escriba los constructores y el destructor de *arbol*. **1,5 p**

b) ¿Es necesario añadir el destructor a *arbolPopio*? ¿Por qué? Si lo es, escríbalo. **0,5 p**

c) ¿Es necesario añadir el constructor de copia a *arbol*? ¿Por qué? Si lo es, escríbalo. **0,5 p**

```
template <class T>
arbol<T>::arbol(): a(0)
{}

template <class T>
arbol<T>::arbol(const T& r):
    a(new arbolPropio<T>(r, arbol(), arbol())) {}

template <class T>
```

```
arbol<T>::arbol(const T& r, const arbol<T>& sai, const arbol<T>& sad):
    a(new arbolPropio<T>(r, sai, sad))
{}

template <class T>
arbol<T>::arbol(const arbol<T> & a)
{
    this->a = a.a ? new arbolPropio<T>(*a.a): 0;
}

template <class T>
arbol<T>::~arbol()
{
    delete a;
}
```

2. Un centro de enseñanza desea gestionar los alumnos matriculados en las diferentes asignaturas que ofrece y los profesores que las imparten. Las restricciones serán las siguientes: **2,5 p**

- Las asignaturas se organizan en grupos de alumnos de tal forma que cada grupo pertenece a una única asignatura.
- Cada alumno pertenece a un grupo de cada asignatura en la que está matriculado.
- Los profesores podrán impartir clase en distintos grupos de la misma o diferentes asignaturas. Cada grupo sólo será impartido por un profesor.

a) Identifique las clases y las relaciones que se establecen entre ellas. Dibuje el diagrama de clases. **1,5 p**

b) Implemente las clases que aparecen, escribiendo exclusivamente los miembros imprescindibles para implementar las relaciones. **1,0 p**

3. Sobrecargue paramétricamente el operador de inserción para objetos del tipo vector de conjuntos del parámetro de tipo. Utilice los contenedores `vector` y `set` de la STL, y la función `copy` (que recibe dos iteradores de entrada y uno de salida, y asigna secuencialmente cada elemento del rango de entrada a su correspondiente en el de salida). **2,5 p**

A continuación escriba un programa de ejemplo. El programa rellenará un vector de conjuntos de enteros desde un fichero. Suponga que cada línea contiene todos los elementos de un mismo conjunto. Seguidamente, utilice el operador anterior para imprimir los elementos almacenados en el contenedor en la salida estándar.

```
#include <iostream>
#include <vector>
#include <set>
#include <fstream>
#include <sstream>
#include <iterator>

using namespace std;

template <typename T>
ostream & operator <<(ostream& fs, const vector<set<T> >& v)
{
    for(typename vector<set<T> >::const_iterator i = v.begin(); i != v.end(); ++i) {
        fs << "{ ";
        copy(i->begin(), i->end(), ostream_iterator<T>(fs, " "));
```

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en ing.es

1/6
Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

Si ya tuviste sufi con tanto estudio...
Te dejamos este espacio
para desahogarte.

Pinta, arranca,
llora... tú decides ;)

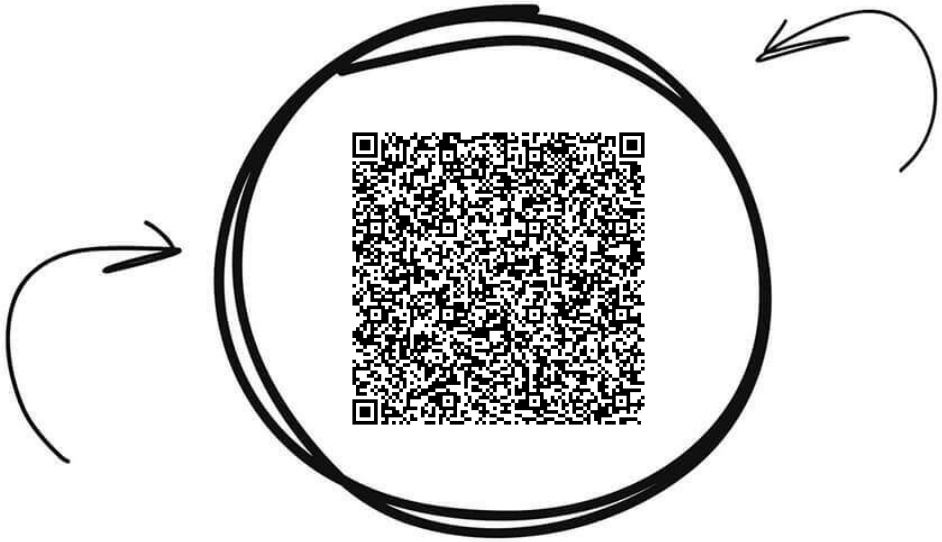
¿Te sientes más liberado?
Sigue siéndolo con la **Cuenta NoCuenta:**
libre de comisiones*, y de lloraditas.

¡Quiero una de esas!

*TIN 0 % y TAE 0 %.

do your thing

Programación Orientada a Obj...



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

```
        fs << "}" << endl;
    }
}

int main()
{
    ifstream fef("datos.dat");
    string cadena;

    vector<set<int> >v;
    while(getline(fef, cadena)) {
        istringstream fec(cadena);
        int e;
        set<int> c;
        while(fec >> e)
            c.insert(e);
        v.push_back(c);
    }
    cout << v;
}
```

4. Sea *Figura* una clase abstracta de la que derivan las clases *Circulo*, *Triangulo*, *Cuadrado* y otras. Supongamos que existe una función para rotar figuras definida como sigue: **1,5 p**

```
void rotar(const Figura& fig)
{
    if (typeid(fig) == typeid(Circulo)) {
        // no hacer nada
    }
    else if (typeid(fig) == typeid(Triangulo)) {
        // rotar trÃangulo
    }
    else if (typeid(fig) == typeid(Cuadrado)) {
        // rotar cuadrado
    }
    // y otras
}
```

- Ponga un ejemplo de uso de la función *rotar*.
- ¿Cree que ésta es la mejor forma de implementar la rotación de figuras? Razone la respuesta. En caso negativo, describa cómo mejorarla y emplee el ejemplo anterior para mostrar la diferencia de uso.

5. Conteste verdadero (V) o falso (F) a las siguientes afirmaciones. Un acierto suma 0,1 y un fallo resta un acierto. **1,0 p**

- En C++ una estructura (struct) sólo puede contener datos, mientras que una clase puede agrupar datos y métodos. (F)
- Un objeto local declarado pero no inicializado queda en un estado indefinido. (F)
- Una clase no puede tener más de un operador de asignación. (F)
- Un destructor no tiene parámetros. (V)
- Un método const puede modificar un atributo mutable. (V)

- El principio de ocultación de información permite que un método observador devuelva una referencia no constante a un atributo privado.(F)
- Una función amiga de una clase C sólo tiene acceso a los miembros públicos de C. (F)
- Una función amiga de una clase C tiene acceso al puntero this. (F)
- El único parámetro de un constructor de copia se puede pasar por valor, aunque a veces es obligatorio pasarlo por referencia. (F)
- El único parámetro del operador de asignación se puede pasar siempre por valor. (V)