Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

2 de septiembre de 2020

Examen de **Tecnología de Programación**

Grado en Ingeniería Informática

Duración total del examen: **1 horas 30 minutos**

NOTA RECORDATORIA. La evaluación final consta de las siguientes pruebas:

Prueba escrita → 60% de la nota (mínimo 5 para aprobar)
Prácticas → 40% de la nota

Ejercicio 1 [1 puntos]

Ejercicio para copiar y pegar imágenes en los enunciados de Moodle.

El siguiente programa en C++ se compila con el comando `g++ main.cc -std=c++11` (gcc versión > 4.8.1). Ignora errores tipográficos y potenciales memory leaks.

¿Compila o no compila? Si no compila ¿Por qué no? Si compila, ¿qué sacaría por pantalla?

foo.h

```
class Foo {
protected:
 int i;

public:
 Foo(int i_):
   i(i_) {}

 virtual void sil()
 {  i*=2;    }

 int meta() const
 {  return i;  }
};
```

bar.h

```
#include "foo.h"
class Bar: public Foo
{
public:
 Bar() :
   Foo(-1) { }

 void sil() override
 {  i+=2;    }

 int meta() const
 {  return i-1; }

};
```

main.cc

```
#include <iostream>
#include "bar.h"
using namespace std;

void mira(Foo* foo) {
   foo->sil();
   cout<<foo->meta();
   cout<<endl;
}

int main() {
  mira(new Foo(1));
  mira(new Bar());
}
```

foo.h

```
class Foo {
protected:
 int i;

public:
 Foo(int i_):
   i(i_) {}

 void sil()
 {  i*=2;    }

 int meta() const
 {  return i;  }
};
```

bar.h

```
#include "foo.h"
class Bar: public Foo
{
public:
 Bar() :
   Foo(-1) { }

 void sil()
 {  i+=2;    }

 int meta() const
 {  return i-1; }

};
```

main.cc

```
#include <iostream>
#include "bar.h"
using namespace std;

void mira(Foo* foo) {
   foo->sil();
   cout<<foo->meta();
   cout<<endl;
}

int main() {
  mira(new Foo(1));
  mira(new Bar());
}
```

foo.h

```
class Foo {
protected:
 int i;

public:
 Foo(int i_):
   i(i_) {}

 void sil()
 {  i*=2;    }

 virtual int meta() const
 {  return i;  }
};
```

bar.h

```
#include "foo.h"
class Bar: public Foo
{
public:
 Bar() :
   Foo(-1) { }

 void sil()
 {  i+=2;    }

 int meta() const
          override
 {  return i-1; }
};
```

main.cc

```
#include <iostream>
#include "bar.h"
using namespace std;

void mira(Foo* foo) {
   foo->sil();
   cout<<foo->meta();
   cout<<endl;
}

int main() {
  mira(new Foo(1));
  mira(new Bar());
}
```

foo.h

```
class Foo {
protected:
 int i;

public:
 Foo(int i_):
   i(i_) {}

 virtual void sil()
 {  i*=2;    }

 virtual int meta() const
 {  return i;  }
};
```

bar.h

```
#include "foo.h"
class Bar: public Foo
{
public:
 Bar() :
   Foo(-1) { }

 void sil() override
 {  i+=2;    }

 int meta() const
          override
 {  return i-1; }
};
```

main.cc

```
#include <iostream>
#include "bar.h"
using namespace std;

void mira(Foo* foo) {
   foo->sil();
   cout<<foo->meta();
   cout<<endl;
}

int main() {
  mira(new Foo(1));
  mira(new Bar());
}
```

foo.h

```
class Foo {
protected:
 int i;

public:
 Foo(int i_):
   i(i_) {}

 virtual void sil()
 {  i*=2;    }

 int meta() const
 {  return i;  }
};
```

bar.h

```
#include "foo.h"
class Bar: public Foo
{
public:
 Bar() :
   Foo(-1) { }

 void sil() override
 {  i+=2;    }

 int meta() const
 {  return i-1; }

};
```

main.cc

```
#include <iostream>
#include "bar.h"
using namespace std;

void mira(Bar* bar) {
   bar->sil();
   cout<<bar->meta();
   cout<<endl;
}

int main() {
  mira(new Foo(1));
  mira(new Bar());
}
```

Ejercicio 2 [1 puntos]

Ejercicio para copiar y pegar imágenes en los enunciados de Moodle.

El siguiente programa en C++ se compila con el comando `g++ main.cc -std=c++11` (gcc versión $>$ 4.8.1). Ignora errores tipográficos y potenciales memory leaks.

¿Compila o no compila? Si no compila ¿Por qué no? Si compila, ¿qué sacaría por pantalla?

foo.h

```
#include <string>
template<typename T>
class Foo {
protected:
 T t;
public:
 Foo(const T& _t) :
    t(_t) {}
 T meto() const {
    return t;
 }
};
```

bar.h

```
#include "foo.h"
template<typename T>
class Bar :
 public Foo<T> {
public:
 Bar(const T& _t) :
    Foo<T>(_t) {}
 T meto() const {
    return this->t+1;
 }
};
```

main.cc

```
#include "bar.h"
#include <iostream>
using namespace std;
template<typename T>
void show(const T& t) {
 cout<<t.meto()<<endl;
}
int main() {
 show(Foo<int>(1));
 show(Bar<int>(1));
 show(Foo<double>(3.0));
 show(Bar<double>(3.0));
}
```

foo.h

```
#include <string>
template<typename T>
class Foo {
protected:
 T t;
public:
 Foo(const T& _t) :
    t(_t) {}
 T meto() const {
    return t;
 }
};
```

bar.h

```
#include "foo.h"
template<typename T>
class Bar :
 public Foo<T> {
public:
 Bar(const T& _t) :
    Foo<T>(_t) {}
 T meto() const {
    return this->t+1;
 }
};
```

main.cc

```
#include "bar.h"
#include <iostream>
using namespace std;
template<typename T>
void show(const T& t) {
 cout<<t.meto()<<endl;
}
int main() {
 show(Foo<int>(1));
 show(Bar<int>(1));
 show(Foo<string>("3"));
 show(Bar<string>("3"));
}
```

foo.h

```
#include <string>
template<typename T>
class Foo {
protected:
 T t;
public:
 Foo(const T& _t) :
    t(_t) {}
 T meto() const {
    return t;
 }
};
```

bar.h

```
#include "foo.h"
template<typename T>
class Bar :
 public Foo<T> {
public:
 Bar(const T& _t) :
    Foo<T>(_t) {}
 T meto() const {
    return this->t+1;
 }
};
```

main.cc

```
#include "bar.h"
#include <iostream>
using namespace std;
template<typename T>
void show(const T& t) {
 cout<<t.meto()<<endl;
}
int main() {
 show(Foo<int>(1));
 show(Foo<float>(2.0f));
 show(Foo<double>(3.0));
 show(Foo<string>("3"));
}
```

foo.h

```
#include <string>
template<typename T>
class Foo {
protected:
 T t;
public:
 Foo(const T& _t) :
    t(_t) {}
 T meto() const {
    return t;
 }
};
```

bar.h

```
#include "foo.h"
template<typename T>
class Bar :
 public Foo<T> {
public:
 Bar(const T& _t) :
    Foo<T>(_t) {}
 T meto() const {
    return this->t+1;
 }
};
```

main.cc

```
#include "bar.h"
#include <iostream>
using namespace std;
template<typename T>
void show(const Bar<T>& t) {
 cout<<t.meto()<<endl;
}
int main() {
 show(Foo<int>(1));
 show(Bar<int>(1));
 show(Foo<double>(3.0));
 show(Bar<double>(3.0));
}
```

Ejercicio 3 [1 puntos]

Ejercicio para copiar y pegar imágenes en los enunciados de Moodle.

El siguiente programa en Haskell se compila con el GHC estándar (ghc Main.hs).

¿Qué hace la función bar? ¿Qué muestra por pantalla el programa principal?

Foo.hs

```
module Foo where
pet :: Integer -> Bool
pet n =
  0 == length
    [x | x <- [2..(n-1)], (n `mod` x) == 0]

foo :: Integer -> [(Integer,Integer)]
foo n = [(x,y) | x <- [2..(n `div` 2)],
  y <- [x..(n-1)], x+y == n, pet x, pet y]
```

Main.hs

```
import Foo
bar :: [Integer]
bar = filter
  ((> 0).length.foo) [1..]

main = do
  print $ take 10 bar
```

Foo.hs

```
module Foo where
pet :: Integer -> Bool
pet n =
  0 == length
    [x | x <- [2..(n-1)], (n `mod` x) == 0]

foo :: Integer -> [(Integer,Integer)]
foo n = [(x,y) | x <- [2..(n `div` 2)],
  y <- [x..(n-1)], x*y == n, pet x, pet y]
```

Main.hs

```
import Foo
bar :: [Integer]
bar = filter
  ((> 0).length.foo) [1..]

main = do
  print $ take 10 bar
```

Foo.hs

```
module Foo where
pet :: Integer -> Bool
pet n =
  n == foldl (+) 0
    [x | x <- [1..(n-1)], (n `mod` x) == 0]

foo :: Integer -> [(Integer,Integer)]
foo n = [(x,y) | x <- [2..(n `div` 2)],
  y <- [x..(n-1)], x+y == n, pet x, pet y]
```

Main.hs

```
import Foo
bar :: [Integer]
bar = filter
  ((> 0).length.foo) [1..]

main = do
  print $ take 10 bar
```

Foo.hs

```
module Foo where
pet :: Integer -> Bool
pet n =
  n == foldl (+) 0
    [x | x <- [1..(n-1)], (n `mod` x) == 0]

foo :: Integer -> [(Integer,Integer)]
foo n = [(x,y) | x <- [2..(n `div` 2)],
  y <- [x..(n-1)], x*y == n, pet x, pet y]
```

Main.hs

```
import Foo
bar :: [Integer]
bar = filter
  ((> 0).length.foo) [1..]

main = do
  print $ take 10 bar
```

Foo.hs

```
module Foo where
pet :: Integer -> Bool
pet n =
  0 == length
    [x | x <- [2..(n-1)], (n `mod` x) == 0]

foo :: Integer -> [(Integer,Integer)]
foo n = [(x,y) | x <- [2..(n `div` 2)],
   y <- [x..(n-1)], x+y == n, pet x, pet y]
```

Main.hs

```
import Foo
bar :: [Integer]
bar = filter
  ((== 0).length.foo) [1..]

main = do
  print $ take 10 bar
```

Foo.hs

```
module Foo where
pet :: Integer -> Bool
pet n =
  n == foldl (+) 0
    [x | x <- [1..(n-1)], (n `mod` x) == 0]

foo :: Integer -> [(Integer,Integer)]
foo n = [(x,y) | x <- [2..(n `div` 2)],
  y <- [x..(n-1)], x+y == n, pet x, pet y]
```

Main.hs

```
import Foo
bar :: [Integer]
bar = filter
  ((== 0).length.foo) [1..]

main = do
  print $ take 10 bar
```

Ejercicio 4 [2 puntos]

Muchos lenguajes de programación modernos permiten programación genérica, en la que el código se aplica a un tipo de dato todavía sin definir. Sin embargo, en ocasiones es necesario imponer restricciones sobre el tipo de dato involucrado. En el campo de texto habilitado en la pregunta, explica brevemente con tus palabras por qué podría ser necesario imponer dichas restricciones.

Enumera las herramientas que ofrecen para declarar e implementar esas restricciones

- un lenguaje orientado a objetos (elige entre C++ y Java)
- un lenguaje funcional (Haskell)

Para ilustrar el funcionamiento de esas herramientas, escribe en ambos lenguajes (uno orientado a objetos y otro funcional), las dos funciones (o métodos) siguientes:

OPCION 1:

- uno que calcule el máximo de los elementos de una colección
- otro que calcule la media de los elementos de una colección

OPCION 2:

- uno que calcule el mínimo de los elementos de una colección
- otro que calcule la suma de los elementos de una colección

OPCION 3:

- uno que calcule si todos los elementos de una colección son iguales
- otro que calcule el producto de los elementos de una colección

Para cada uno de ellos escribe un pequeño programa principal que demueste su uso con una colección válida de datos y con una no válida (de forma que de un error de compilación en el segundo caso). Explica para cada lenguaje y para cada método o función por qué uno de los dos usos del programa principal compila y funciona y por qué el otro caso da un error de compilación.

Los ficheros fuente que generes se empaquetarán en un **único archivo** llamado

ejercicio4_<NIP>.zip

que se entregará como fichero adjunto a esta pregunta.

Para el cuadro de texto:

Explica con tus palabras por qué podría ser necesario imponer restricciones sobre el tipo de dato en programación genérica.

Enumera las herramientas que ofrecen para declarar e implementar esas restricciones

- un lenguaje orientado a objetos (elige entre C++ y Java)
- un lenguaje funcional (Haskell)

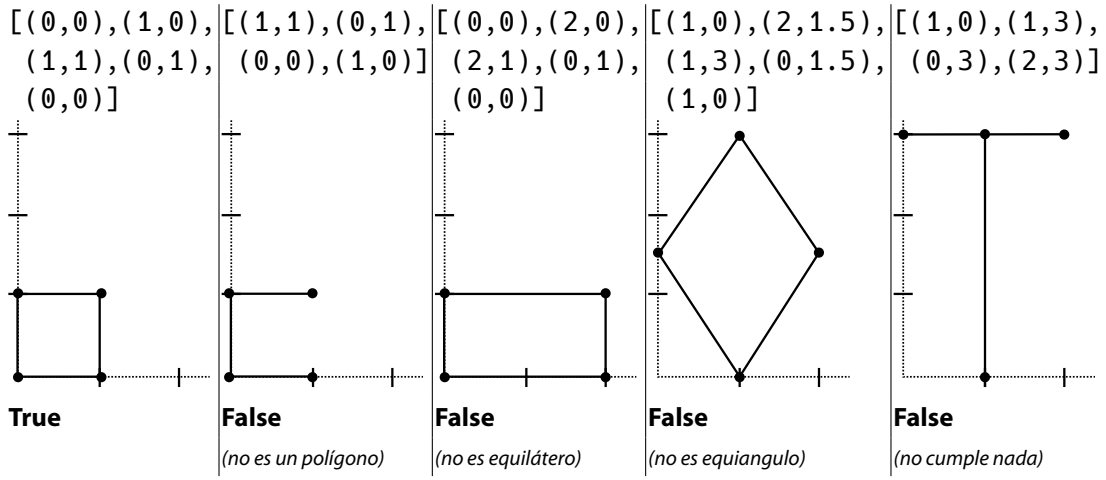
Explica para cada lenguaje y para cada método o función por qué uno de los dos usos del programa principal compila y funciona y por qué el otro caso da un error de compilación.

Ejercicio 5 [2 puntos]

Una herramienta de diseño CAD permite modelar polilíneas en base a **secuencias de vértices**. Cada vértice se representa mediante una tupla con sus dos coordenadas reales, y por tanto cada polilínea se define como una lista de tuplas (vértices).

Se pide definir una función `esPoligonoRegular` que devuelva si una polilínea es un **polígono regular** o no. Para ello hay que hacer las siguientes comprobaciones:

- Si la polilínea es un **polígono**, es decir, si el último vértice coincide con el primero.
- Si el polígono es **equilátero**, es decir, si todas sus aristas (rectas que unen vértices contiguos) tienen la misma longitud. La longitud se mide con la distancia euclídea, es decir, dados dos vértices con coordenadas $\mathbf{v_0} = (x_0, y_0)$ y $\mathbf{v_1} = (x_1, y_1)$ la arista $\mathbf{a_{01}} = \mathbf{v_1} - \mathbf{v_0} = (x_{01}, y_{01}) = (x_1 - x_0, y_1 - y_0)$ tendrá longitud $|\mathbf{a_{01}}| = \sqrt{x_{01}^2 + y_{01}^2}$.
- Si el polígono es **equiángulo**, es decir, si los ángulos que forman todos los pares contiguos de aristas son idénticos. Para calcular el ángulo hay que aplicar las dos definiciones de producto escalar $\mathbf{a_{01}} \cdot \mathbf{a_{12}} = x_{01}x_{12} + y_{01}y_{12} = |\mathbf{a_{01}}||\mathbf{a_{12}}| \cos\alpha$ donde α es el ángulo que forman las dos aristas (que debe ser despejado de la ecuación). Se debe tener en cuenta que la última arista y la primera arista también forman un ángulo que debe ser comprobado.



Implementa dos versiones de la misma función:

- `esPoligonoRegular` que sea lo más **breve y concisa** que puedas, utilizando los elementos del lenguaje que sean necesarios (como funciones de orden superior).
- `esPoligonoRegular'` que se recorra la lista de tuplas **una única vez**, aunque su código sea inevitablemente más largo.

Se permite definir funciones auxiliares adicionales si lo consideras necesario. Se puede utilizar cualquier función del `Prelude` de Haskell, pero no otras funciones del lenguaje.

Ejercicio 6 [3 puntos]

Una editorial de comics digitales desea gestionar automáticamente su biblioteca. Desea utiliar un programa que almacene en una estructura de datos la información de los comics, con las siguites condiciones:

- Cualquier **Comic** se considera que pertenece a una **Saga** (conjunto de historias con un conjunto de personajes común y un arco argumental).
- Dentro de una **Saga**, un **Comic** puede ser un **Volumen** indepediente o un **Episodio** dentro de una Serie con una historia concreta común.
- La **Biblioteca** está formada por un conjunto de Sagas.

Más concretamente:

- Un Comic y un Volumen contienen información sobre su año de publicación (un entero) y sobre la Saga a la que pertenece (cadena de caracteres).
- Un Episodio contiene además información sobre la Serie a la que pertenece (cadena de caracteres).
- Una Serie almacena un conjunto de Episodios (no Volumenes), ordenados por su año de publicación. Almacena también un año de publicación, que se considera igual al del primer Episodio de la Serie, y la Saga a la que pertenece.
- Una Saga almacena un cojunto (sólo UNO) de elementos que pueden ser bien un Volumen independiente o una Serie. Guarda también un año de publicación, que es el del primer elemento de la Saga.

Implementa en el lenguaje de tu elección (C++ o Java) la siguiente funcionalidad:

- Define el conjunto de clases necesario para representar la información descrita en el problema.
- Un constructor de la Biblioteca, a partir de una lista de Comics:

  ```
  Biblioteca ( <conjunto de Comics> )
  ```

  que genera la lista de Sagas que contiene la Biblioteca. Ello implica añadir el Comic a su Saga si es un Volumen, o a una Serie dentro de su Saga correspondiente si es un Episodio, creando ambas si es necesario o usando una ya existente si procede.
- Un método en la clase Biblioteca

  ```
  <lista de elementos> buscaAteriores(... año);
  ```

  que devuelva el conjunto de Volumenes y Series publicadas hasta ese año (inclusive).

Para optimizar las búsquedas y mantener el año de publicación de una Saga o una Serie correctamente actualizado, cada vez que se añada un elemento a una de ellas, el contenido correspondiente deberá quedar ordenado por año de publicación, y el año correspondiente de la Saga o Serie actualizado.

Para ello puedes implementar los métodos que consideres oportunos en el resto de clases. Se valorará la no repetición de código.

Puedes utilizar toda la funcionalidad definida en la biblioteca estándar de C++14 o Java.