ESTRUCTURAS DE DATOS Y ALGORITMOS

1º de Grado en Ingeniería Informática

(Curso 2021-2022)

1 PRÁCTICA 0

1. NORMAS GENERALES

1. Los alumnos del Doble Grado **en Ingeniería Informática y Business ANALYTICS** deberán consultar también el documento ANEXO PRÁCTICA 0.
2. El código deberá mostrar modularidad, calidad, legibilidad y el uso apropiado de estructuras y la inclusión de comentarios.
3. Los alumnos del Grado en Informática, en esta práctica deberán codificar tanto en C como en C#. Los alumnos del Doble Grado solo en C, y deberán realizar el ejercicio alternativo a C# indicado en el documento ANEXO PRÁCTICA 0.
4. El alumno deberá subir a la plataforma virtual la práctica realizada. La subida al espacio habilitado podrá realizarse hasta el 26/09/2021 en convocatoria ordinaria y hasta el 29/05/2022 en convocatoria extraordinaria, ambas fechas inclusive.
5. Se deberá subir únicamente un fichero *.rar, nombrado del siguiente modo: PR0_NombreAlumno1ApellidoAlumno1NombreAlumno2ApellidoAlumno2.rar (Ejemplo: PR0_AntonioPerezLuisaJimenez.rar). El nombre y el apellido del alumno no deberá contener acentos.
6. El fichero *.rar contendrá:
    - En el caso de C únicamente los ficheros fuente, nombrados siguiente modo: PR0_P1_NombreAlumno1ApellidoAlumno1NombreAlumno2ApellidoAlumno2.c (Ejemplo: PR0_P1_AntonioPerezLuisaJimenez.c). El nombre y apellido del alumno, no deberán contener acentos.

- En el caso C# todo el proyecto de Visual Studio.
- Un fichero *.docx con el nombre y apellidos de los alumnos implicados en la práctica.

7. Aquellos ejercicios cuyos ficheros de entrega no cumplan la normativa de nombrado se calificarán con 0 puntos.
8. El ejercicio deberá realizarse en equipo. **El equipo será de dos personas**. Si el número de alumnos de clase fuese impar podrá haber un único grupo de tres alumnos, que deberán realizar adicionalmente el ejercicio descrito en el documento Anexo_Práctica0_EjercicioAdicional. No se permiten otras configuraciones de grupos.

2. OBJETIVOS

- El objetivo de esta práctica es que el alumno recuerde algunos de los conceptos básicos de programación, y que ya debe conocer, tras haber cursado las asignaturas Introducción a la Programación (IP) y Programación Orientada a Objetos (POO).

3. ENUNCIADO

3.1 Primera parte: Fundamentos de C

3.1.1. Programa 1: Función MiPotencia()

Escribir una función MiPotencia() que calcule la potencia de un número real elevado a un número entero positivo (sin hacer uso de la función pow() de math.h)

Escribir un programa principal que lea de teclado la base, el exponente (y lo valide: el exponente debe ser positivo) y a continuación que llame a la función MiPotencia()

float MiPotencia(float base, int e);

Después el programa principal mostrará el resultado.

3.1.2. Programa 2: Función Serie()

Escribir una función Serie() que calcule la siguiente serie:

1 - x + x2 - x3 + x4 + ...

Para ello, dentro de la función Serie() , para calcular cada término, se debe llamar a la función anterior MiPotencia().

Prototipo de Serie():

float Serie(float x, int nt);

Escribir el programa principal, que leerá el valor de x, así como el número de términos a sumar de la serie (validando que sea positivo)

A continuación, el programa principal debe llamar a la función Serie(), y después mostrará el resultado por pantalla.

3.1.3. Programa 3: Función void

Escribir un programa que lea del teclado la altura de un triángulo (validar que es positiva) y llama a la función DibujarTriángulo() que lo dibuja por pantalla. Se utilizan dos funciones void (no devuelven ningún valor): La función (DibujarTriángulo()) dibujará el triángulo, que a su vez llamará a otra función DibujarLinea(), para dibujar cada línea de asteriscos.

Por ejemplo, el triángulo de altura 5 debe aparecer de la siguiente manera:

*****

****

***

**

*

3.2. Segunda parte: Fundamentos de POO

En este ejercicio van a poner en práctica algunos de los conocimientos adquiridos en programación orientada a objetos. Se codificará en C#. Específicamente, el concepto de herencia, clases y métodos abstractos, métodos virtuales y de reemplazo, etc. Se debe codificar un programa en C# que presente el siguiente menú de opciones:

1. Sumar dos vectores de 2 dimensiones en coordenadas cartesianas. Esta opción pide la entrada de coordenadas cartesianas de los vectores de dos dimensiones por pantalla, realiza la suma y muestra el resultado en pantalla en coordenadas cartesianas.
2. Sumar dos vectores de 3 dimensiones en coordenadas cartesianas. Esta opción pide la entrada de coordenadas cartesianas de los vectores de tres dimensiones por pantalla, realiza la suma y muestra el resultado en pantalla en coordenadas cartesianas.

3. Sumar dos vectores de 2 dimensiones en coordenadas polares. Sumar dos vectores de 2 dimensiones en coordenadas polares. Esta opción pide la entrada de coordenadas polares de los vectores de dos dimensiones por pantalla, realiza la suma y muestra el resultado en pantalla en coordenadas polares.
4. Sumar dos vectores de 3 dimensiones en coordenadas esféricas. Sumar dos vectores de 3 dimensiones en coordenadas esféricas. Esta opción pide la entrada de coordenadas polares de los vectores de tres dimensiones por pantalla en coordenadas esféricas, realiza la suma y muestra el resultado en pantalla en coordenadas esféricas.

Deberá definirse y utilizarse la siguiente clase abstracta:

```
abstract class Vectores

{

      public abstract void ImprimirPorPantalla(double[] vector);
      public abstract double[] SumarVectores(double[] vector1, double[]
vector2);
      public abstract double[] LeerVector(ref double[] vector);


}
```

Así como las siguientes clases hijas de dicha clase principal:

```
// Clase para vectores de 2 dimensiones en coordenadas cartesianas

class Vec2DC : Vectores

{

}

// Clase para vectores de 2 dimensiones en coordenadas polares

class Vec2DP : Vectores
{

}

// Clase para vectores de 3 dimensiones en coordenadas cartesianas
```

```
class Vec3DC : Vectores

{

}



}

//Clase para vectores de 3 dimensiones en coordenadas esféricas

class Vec3DE : Vectores

 {

}
```

Las clases hijas emplearán el polimorfismo, implementando en ellas los métodos ImprimirPorPantalla, SumarVectores y LeerVector.