Informática Básica

2022-2023

Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

1	2	3	4	5

6 de julio de 2023

Apellidos: **Nombre:**

DNI: **Turno (M/T):** **e-mail:**

Lea detenidamente estas instrucciones:

- No escriba nada en las casillas de la parte superior de esta página.
- El examen consta de 5 preguntas y el tiempo disponible es de 3 horas.
- Realice cada ejercicio con bolígrafo en un FOLIO DISTINTO y entregue los ejercicios en orden.
- Ponga nombre, apellidos y DNI en todos los folios que utilice.
- Al finalizar el examen, ENTREGUE TODOS LOS FOLIOS que haya utilizado, incluyendo éste.
- Al entregar el examen muestre algún documento que acredite su identidad.

---

1. **1.75 Puntos**. Bases de datos no relacionales. Explique (no más de un folio escrito por ambas caras) (a) Qué son este tipo de bases y cuales son sus características definitorias. (b) Exponga cuáles son los tipos de bases de datos no relacionales más habituales.

2. **1.75 Puntos**. Describa detalladamente qué es la sobrecarga de funciones, qué uso tiene e ilustre su exposición con un ejemplo representativo.

3. **2 Puntos**. Desarrolle un programa que, dada una cadena de caracteres, determine si la cadena corresponde con un número de tarjeta de crédito válida, utilizando el algoritmo de Luhn. En ese algoritmo las cadenas de longitud uno o menor no son válidas. Se permite que la cadena contenga espacios, pero en ese caso han de ser eliminados. La cadena solo puede contener caracteres correspondientes a dígitos (0, … 9).

Considérese la cadena "`4539 3195 0343 6467`". El primer paso del algoritmo consiste en duplicar cada dos dígitos, empezando por la derecha, de modo que en este ejemplo se doblarán los dígitos en negrita: **4**5**3**9**3**1**9**5**0**3**4**3**6**4**6**7 Si al doblar un dígito, resulta un valor mayor que 9 (p. ej. 6x2=12), entonces sume los dígitos del producto (p. ej. 12: 1+2=3). El resultado del doblado sería: `8569619503833437`. A continuación se suman todos los dígitos: `8+5+6+9+6+1+9+5+0+3+8+3+3+4+3+7=80` Si el resultado es múltiplo de 10, el número es válido, en caso contrario no.

4. **2.5 Puntos**. Se desea crear una clase *DoubleManager* que almacene un conjunto de números reales, del que se quiere poder encontrar valores próximos entre sí. Se propone la siguiente estructura de datos: la clase *DoubleManager* contendrá como atributo un array de vectores, de forma que las coordenadas de cada punto lo ubiquen en un vector concreto dentro de dicho array. De esta forma, los puntos próximos entre sí se almacenarán en un mismo vector.

Para este ejercicio, se asume que los valores a almacenar están dentro del intervalo [0.00, 9.99] y que el array a utilizar tiene tamaño 10. Para obtener a qué vector corresponde un valor concreto, basta convertirlo a entero (de *double* a *int*).

Diseñe un programa en el que se defina la clase *DoubleManager*, que contenga un array *grid_* de 10 vectores y los siguientes métodos:

- Un método privado *GetIndex* que reciba como parámetro un valor de tipo *double* y devuelva su índice entero correspondiente. El método debe aseverar (*assert*) que el índice a devolver está en el intervalo correcto [0, 9] antes de devolverlo.
- Un método público *Add* que recibe como parámetro un valor de tipo *double* a insertar en el vector que le corresponda dentro del array.
- Un método public *Closest* que recibe como parámetro un valor de tipo *double* y devuelve el valor almacenado en el array que se encuentre más cerca del mismo. Para ello, el método obtendrá el índice del vector correspondiente al valor dado y explorará dicho vector *y sus adyacentes* (es decir, para un punto ubicado en el vector *i*, se analizará desde (*i* - 1) hasta (*i* + 1), siempre y cuando estos índices no se salgan del array). Si durante la búsqueda se encuentra un valor que sea igual que el que se ha dado por parámetro (es decir, cuya diferencia en valor absoluto sea menor que un cierto valor constante *epsilon*), debe ignorarse y pasar al siguiente. Si no se encuentra ningún valor cercano, el método devolverá el valor -1. Por simplicidad, se puede asumir que la distancia máxima aceptable entre valores vecinos es de 1.0 (es decir, la distancia máxima entre dos valores que pertenecen al mismo vector).

Por último, en la función *main* del programa cree un *DoubleManager*, introduzca los valores 3.9, 4.1 y 4.9, y obtenga en una variable local cuál es el valor más cercano a 4.1. Indique, según su propia implementación, cuál debería ser el resultado. No es necesario que el programa muestre nada por pantalla. Se valorará la limpieza, la documentación y la eficiencia del código.

5. **2 Puntos**. Responda en este mismo folio Verdadero (V) o Falso (F) para cada una de las siguientes preguntas teniendo en cuenta las siguientes observaciones:

- Por cada respuesta correcta se sumará 0.2 puntos.
- Por cada respuesta incorrecta se restará 0.2 puntos.
- Las preguntas sin responder se considerarán incorrectas.
- La puntuación mínima de la pregunta es de 0 puntos.

1.- El comando de Linux que permite mostrar en pantalla su nombre de usuario es `uname`

2.- El preprocesador de C++ elimina del código los comentarios que el usuario puede haber escrito

3.- De acuerdo a la Guía de Estilo de Google para C++ los identificadores de funciones ordinarias (no métodos de una clase) deben escribirse comenzando con letra mayúscula y usando mayúscula para cada palabra.

4.- La salida que el siguiente programa imprime en pantalla es: `2 11`

```
#include <iostream>

void Sum(int value_a, int value_b, int& value_c) {
  value_a = value_b + value_c;
  value_b = value_a + value_c;
  value_c = value_a + value_b;
}

int main() {
  int first{2}, second{3};
  Sum(first, second, second);
  std::cout << first << " " << second;
  return 0;
}
```

5.- El método `push_back()` permite añadir elementos a un objeto de la clase `std::vector`

6.- Al leer caracteres desde un flujo de entrada (`cin`), la lectura se detiene al detectar un punto

7.- Los atributos de una clase sólo pueden ser públicos o privados

8.- Cualquier operador puede ser sobrecargado mediante una función `friend`

9.- Una función se puede declarar `friend` de más de una clase.

10.- En una relación de composición (entre objetos de diferentes clases), la parte (miembro) desconoce la existencia del objeto