Universidad de Granada
Departamento de Ciencias de la Computación
e Inteligencia Artificial

Fundamentos de Programación (2016/17)
1º Grado en Ingeniería Informática
Conv. Extraordinaria - 12 de Julio de 2017

DECSAI

Normas para la realización del examen: **Duración: 2.5 horas**

- **El único material permitido durante la realización del examen es un bolígrafo azul o negro.**
- **Debe disponer de un documento oficial que acredite su identidad a disposición del profesor.**
- **No olvide escribir su nombre completo y grupo en todos y cada uno de los folios que entregue.**

◁ Ejercicio 1 ▷ De bits a char [2 puntos]

Implemente un **programa** que lea valores `int` desde teclado y que tenga en cuenta únicamente los valores `0` y `1` (representan valores de *bits*). El programa terminará cuando se lea un valor negativo. Cada 8 valores de *bits* leídos, el programa calculará el número entero que representa y lo transformará en el carácter (`char`) correspondiente. El primer *bit* leído es el más significativo. Si el carácter corresponde a una letra -mayúscula o minúscula- lo mostrará por pantalla. Finalmente, el programa debe mostrar el porcentaje de letras y otros símbolos leídos.

Si no es posible completar el último bloque con 8 bits, no se considerarán los números leídos.

```
Entrada:  0 1 0 0 1 3 1 1 1 2 0 1 1 0 1 0 1 1 9 0 0 1 7 0 0 0 0 1 1 0 1 -1
Salida: Ok
LETRAS: 66.67%
OTROS:  33.33%
```

Donde las correspondencias son $01001111 \rightarrow$ `O`, $01101011 \rightarrow$ `k`, y $00100001 \rightarrow$ `!`. Los *bits* finales 101 no se consideran.

◁ Ejercicio 2 ▷ Secuencia Cifrada [2 puntos]

Se desea implementar una clase que permita cifrar y descifrar cadenas de caracteres.

Para el cifrado se empleará el *algoritmo de rotación*. Consiste en seleccionar una `clave` (un número entero), y desplazar las letras del alfabeto tantas posiciones como indica la clave. Se considera una representación *circular* del alfabeto, de tal forma que el carácter que sigue a `'Z'` es `'A'`. Por ejemplo, si `clave=4`, entonces la `'A'` se reemplaza por la `'E'` y la `'Z'` por la `'D'`. Utilizando `clave=0` la secuencia cifrada es igual a la original.

Defina una **clase** `Mensaje` que permita almacenar una secuencia de caracteres (almacenados en un *array* de datos `char`), su tamaño máximo, tamaño actual y una clave.

1. Solamente se deben almacenar carácteres que correspondan a letras mayúsculas,
2. El valor de la clave estará comprendido entre 0 y el número de caracteres entre `'A'` y `'Z'` (ambos incluidos).

Proporcione la **implementación** completa de la clase. Deberá incluir, necesariamente:

1. Constructor que recibe un `string` (puede contener caracteres no válidos) y la clave.
2. Método `Cifrar`: para cifrar la secuencia. Modifica el objeto.
3. Método `Descifrar`: para descifrar la secuencia. Modifica el objeto.

Implemente un **programa** que permita cifrar y descifrar un texto. El programa leerá el mensaje desde el teclado en una variable de tipo `string` (puede contener símbolos inválidos) y la clave.

A continuación, usando un menú, preguntará al usuario si desea cifrar o descifrar el mensaje, o terminar. Después de cada operación mostrará por pantalla el resultado de realizar la operación.

◁ Ejercicio 3 ▷ clase `TrackCoordenadaGPS` [3 puntos]

El sistema de posicionamiento global, más conocido por sus siglas en inglés, **GPS** (**G**lobal **P**ositioning **S**ystem), es un sistema que permite determinar en toda la Tierra la posición de un objeto.

Un dispositivo GPS es capaz de captar y registrar la posición en el espacio en base a tres coordenadas: *latitud* y *longitud* (grados) y *altura* (metros). Los valores de latitud y longitud verifican $-90 \leq \text{lat} \leq 90$ y $-180 < \text{lon} \leq 180$.

Suponga que dispone de la clase `CoordenadaGPS` para poder trabajar con posiciones en el espacio.

Escriba en C++ la clase `TrackCoordenadaGPS` que permita almacenar y gestionar una secuencia de datos de tipo `CoordenadaGPS` que representa una sucesión de posiciones sobre la Tierra (representadas por sus coordenadas GPS).

CoordenadaGPS
- double latitud - double longitud - double altura
+ CoordenadaGPS (double la_latitud, double la_longitud, double la_altura) + CoordenadaGPS (void) + void SetCoordenadas (double la_latitud, double la_longitud, double la_altura) + double GetLatitud (void) / + double GetLongitud (void) / + double GetAltura (void) + double DistanciaSobrePlano (CoordenadaGPS otro) + double DistanciaReal (CoordenadaGPS otro) + double DiferenciaAltura (CoordenadaGPS otro)

Defina los datos miembro necesarios para definir dicha clase, e implemente el(los) constructor(es), gestores de la secuencia (añadir objetos de tipo `CoordenadaGPS`, obtener un elemento, obtener el número de datos `CoordenadaGPS` de la secuencia, ...) y métodos que permitan calcular:

1. **Desnivel acumulado positivo** del camino. Suma total de las diferencias de altura **positivas** (subidas) entre cada dos puntos consecutivos.
2. **Longitud real** del camino. Suma total de las distancias (reales) entre cada dos puntos consecutivos.
3. La **máxima distancia (sobre el plano)** entre dos puntos consecutivos.

◁ Ejercicio 4 ▷ clase `FlujoMensajes` [3 puntos]

Una matriz T registra el flujo de mensajes entre un grupo de usuarios de una red social. La componente T_{ij} almacena el número de mensajes enviados por el usuario i al usuario j. El valor T_{ji} se interpreta como el número de mensajes recibidos por el usuario i desde el j (observe que la matriz no tiene porqué ser simétrica). Se define la *actividad* de un usuario i como la cantidad de mensajes que ha enviado. Se define la *popularidad* de un usuario i como la cantidad de mensajes que ha recibido.

La matriz T se puede representar con la clase `TablaRectangular` cuya especificación básica es la siguiente:

TablaRectangular
- static const int NUM_FILS - static const int NUM_COLS - int matriz_privada[NUM_FILS][NUM_COLS] - int filas_utilizadas, cols_utilizadas
+ TablaRectangular() + TablaRectangular(int num_filas, int num_columnas) + int FilasUtilizadas() + int ColumnasUtilizadas() + int Elemento(int fila, int columna) + void Modifica(int fila, int columna, int valor)

Defina una clase `FlujoMensajes`, que contendrá como datos miembro:

```
TablaRectangular actividad;
int num_usuarios;
```

e implemente los siguientes métodos:

1. `int Actividad(int i)`, que calcula la **actividad** de un usuario i
2. `int Popularidad(int i)`, que calcula la **popularidad** de un usuario i
3. `TablaRectangular Resumen()`, que construye y devuelve una nueva `TablaRectangular` de tamaño (útil) `num_usuarios` $\times 2$ (cada fila corresponde a un usuario, la columna 0 almacena su actividad y la 1, su popularidad).
4. `int MasPopular()` que devuelve el **índice** del usuario más popular.