Declaraciones e Implementaciones. Principio de ocultamiento de la información en estructuras de datos y funciones.

Preguntas iniciales

Imagine un programador, que tiene que resolver un problema cualquiera que requiera desarrollar algoritmos y estructuras de datos. Responda:

1. Cuando el programador escribe la definición de una estructura de datos: ¿en que debe focalizarse?
2. Cuando escribe la declaración de una función ¿en que debe focalizarse?
3. Cuando escribe la implementación de una estructura de datos ¿en que debe focalizarse?
4. Cuando escribe la implementación de una función ¿en que debe focalizarse?
5. ¿Qué ventajas tiene pensar en separado la declaración y la implementación?

Estructuras de datos

1) Definir una estructura de datos que represente cualquier carta de una baraja española.
2) Definir una estructura de datos que represente una baraja de cartas españolas.
3) Definir una estructura de datos que represente un número complejo
4) Definir una estructura de datos que represente un planeta y otra que represente una estrella.
5) Definir una estructura de datos que represente el sistema solar
6) Definir una estructura de datos que represente el almanaque de un determinado año.
7) Definir una estructura de datos que represente la hora como un reloj que marca las horas, minutos y segundos.
8) Definir una estructura de datos que permita representar un recordatorio a un día del almanaque. Ejemplo de un recordatorio: *El día Jueves 8 de Septiembre no olvidar ir al dentista*.
9) Definir una estructura de datos que permita parsear una cadena de caracteres en formato CSV
10) Definir una estructura de datos que permita parsear una cadena de caracteres en formato JSON
11) Definir una estructura de datos que permita parsear una cadena de caracteres en formato XML
12) Definir una estructura de datos que represente cualquier pieza de Ajedrez
13) Definir una estructura de datos que represente un jugador de ajedrez que tiene piezas de un mismo color y piezas que fueron comidas por su rival

Declaración de funciones.

Para cada declaración de la función se pide que escriba la pre-condición y la post-condición de la misma. **No se pide que la implemente (solo escriba la declaración con su pre y post condición)**.

1. Una función que determine si una carta es de Espada o no es de Espada.
2. Una función que determine si una carta es de un “palo” determinado o no.
3. Una función que obtenga la suma de dos números complejos.
4. Una función que obtenga todas las cartas de un palo determinado de una baraja española de cartas.
5. Una función que mezcle una baraja española de cartas.
6. Una función que determine si un **planeta** pertenece al **sistema solar**.
7. Una función que cuente la cantidad de caracteres de una cadena de caracteres.
8. Una función que determine si dos cadenas de caracteres son iguales.
9. Una función que determine si una cadena de caracteres está contenida al comienzo de otra cadena de caracteres. Ejemplo: la cadena ‘par’ está contenida al comienzo de la cadena ‘parlamo’
10. Una función que determine la cantidad de apariciones que una cadena de caracteres está contenida en cualquier parte de otra cadena de caracteres. Ejemplo: la cadena “la” aparece 2 veces en la cadena “cala pala”.
11. Una función que devuelva el indice donde aparece una cadena dentro de otra, si esa otra cadena no esta que devuelva el indice donde esta el barra cero
12. Una función que permita cargar una estructura de datos para una cadena en formato CSV
13. Una funcion que inicialice las piezas de un jugador de ajedrez
14. Una funcion que indique cuantas piezas vivas tiene un jugador de ajedrez
15. Una funcion que indique cuantas piezas muertas tiene un jugador de ajedrez
16. Una funcion que le mate una pieza a un jugador de ajedrez

Implementaciones de tipos de datos

Para cada uno de los siguientes ejercicios se pide: Definir la o las estructuras de datos, declarar e implementar las operaciones (funciones). Para cada función se pide que escriba su pre-condición y su pos-condición

1. Defina el tipo de datos NumeroComplejo con las siguientes operaciones
    a. Suma de dos números complejos
    b. Resta de dos números complejos
    c. Multiplicación de dos números complejos
    d. Módulo de un número complejo
2. Defina el tipo de dato Registro (el cual contiene Campos) que permita las siguientes operaciones
    a. Inicializar el tipo de dato Registro apartir de una cadena de caracteres en formato CSV
    b. Obtener la cantidad de campos que posee un Registro

    c. Obtener un campo determinado
3. Defina el tipo de dato Punto que representa el par de valores x e y de un eje cartesiano, donde x e y pueden ser números reales, con las siguientes operaciones
    a. Creación de un Punto
    b. Distancia entre dos puntos
4. Basado en el ejercicio de conectividad de la primer clase se pide crear las siguientes operaciones
    a. La operación que permite saber si p y q están conectados
    b. La operación que permite conectar a p y q
    c. Hacer dichas implementaciones para las estructuras de datos:
        i. Basados en matriz
        ii. Basados en un vector
        iii. Basados en un arbol representado en una estructura de datos vector
        iv. Basados en un arbol balanceado representado en una estructura de datos vector con otro vector de pesos

Implementaciones - Ejercicios de Cadena de caracteres

Restricciones para todos los ejercicios de cadena de caracteres:

- No se puede utilizar ninguna librería auxiliar, solamente iostream. (no incluir stdlib, no utilizar string)
- En los programas de prueba que realice no debe recibir datos de la entrada estándar, los datos de prueba deben estar escritos en el programa main.
- Utilizar memoria estática para las cadenas de caracteres. Esto significa que debe utilizar una constante para determinar el tamaño de todas las cadenas de caracteres que utilice.

1) Implementar una función que cuente la cantidad de caracteres de una cadena de caracteres y devuelva ese valor. Luego:
    a) Escribir un programa de prueba que muestre por la salida estandard varias pruebas.
    b) ¿que pre-condición tiene la función implementada?
    c) Enuncie la post-condición de la función.
    d) Luego de enunciar la post-condición verifique que el nombre de la función y el nombre de los parámetros refleje la misma, si no es así modifique los nombres.
2) Implementar una función que reciba por parámetro dos cadenas de caracteres y que devuelva true si son iguales, de lo contrario devuelve false. Luego
    a) Escribir un programa de prueba que muestre por la salida estandard varias pruebas.
    b) ¿que pre-condición tiene la función implementada?
    c) Enuncie la post-condición de la función.
    d) Luego de enunciar la post-condición verifique que el nombre de la función y el nombre de los parámetros refleje la misma, si no es así modifique los nombres.

3) Implementar la siguiente función: void Copiar(char origen[], char destino[]) que realiza una copia de la cadena origen en la cadena destino. Luego
    a) Escribir un programa de prueba que muestre por la salida estandard varias pruebas.
    b) ¿qué pre-condición tiene esta función?
    c) ¿qué validaciones debería considerar esta función? Implementar esas validaciones y agregar en el programa de pruebas casos de prueba sobre esas validaciones
4) Sobre esta función: void Reemplazar(char cadena[], char cadenaBusqueda[], char cadenaReemplazo[]), la cual tiene la responsabilidad de reemplazar las apariciones en cadena de cadenaBusqueda por cadenaReemplazo. Ejemplos: si se invoca del siguiente modo:

```
char cadena[Tope] = "ASTS"
Reemplazar(cadena, "S", "UY");
```

Se espera que cadena sea: "AUYTUY".
Notar que cadena tiene que tener un tamaño muy grande (dado por una constante) para que en esta cadena se pueda reemplazar.
    a) ¿qué validaciones debería considerar esta función?
    b) ¿Como haría la descomposición funcional de la responsabilidad de esta función?
    c) Implemente un programa que realice varias pruebas sobre esta función.
5) Implementar la siguiente función que cumpla con la post condición indicada:

```
/*
 * Precondición: @cadena es una cadena de caracteres
 * Postcondición: Si @cadena tiene espacios en blancos consecutivos,
 * reduce todas estas apariciones consecutivas de @cadena en un
 * espacio en blanco
 * Ejemplo:
 * para la cadena:
 * "La  noche se astilló  de    estrellas mirándome..."
 * es modificada del siguiente modo:
 * "La noche se astilló de estrellas mirándome..."
 */
void QuitarEspaciosRedundantes(char cadena[]);
```

Implementaciones - Ejercicios de “Zipic”

Definiciones de Zipic

Se define un Zipic como un vector de enteros donde cada elemento del vector es un número entero mayor que cero y la marca de fin de un vector Zipic está dada por el número -1.
Ejemplos:

- [234 12 4 -1] es un Zipic de tamaño 3
- [1 4 -1 -221 0] es un Zipic de tamaño 2. No importa el valor de los elementos luego de la marca de fin (-1).
- [-8 12 4 -1] no es un Zipic (los elementos tienen que ser mayor que cero y el primer elemento no lo es)
- [234 12 4] no es un Zipic porque no esta la marca de fin.

Sobre los Zipic se definen las siguientes operaciones

Sumatoria de un Zipic: Es una operacion que se realiza sobre un Zipic y consiste en realizar la sumatoria de los elementos del Zipic donde cada **elemento** se multiplica por su **posición**.
Ejemplos de sumatoria de Zipic

- La sumatoria de [**234 12 4 -1**] es **270**: (**234** * 1) + (**12** * 2) + (**4** * 3) = **270**
- La sumatoria de [**1 4 -1 -221 0**] es **9**: (**1** * 1) + (**4** * 2) = **9**

Suma entre 2 Zipic: Es una operación que se realiza sobre dos Zipic y consiste en sumar las sumatorias multiplicadas por el tamaño respectivo de cada Zipic
Ejemplos de suma entre Zipic

- [**234 12 4 -1**] + [**1 4 -1 -221 0**] es **828** puesto que: 270 * 3 + 9 *2 = **828**

Multiplicación entre Zipic: Esta operacion se puede realizar entre dos zipic de distintos tamaños.
Ejemplos de multiplicación entre Zipic

- [**234 12 4 -1**] + [**1 4 -1 -221 0**] es puesto que: (**234 * 1**) * 1 + (**12 * 4**) * 2 + **4** * 3

Ejercicios sobre Zipic

1) Implementar una función que determine si un vector de enteros es un Zipic o no. Luego:
    a) Escribir un programa de prueba que muestre por la salida estandard varias pruebas.
    b) ¿que pre-condición tiene la función implementada?
    c) Enuncie la post-condición de la función.
    d) Luego de enunciar la post-condición verifique que el nombre de la función y el nombre de los parametros refleje la misma, si no es así modifique los nombres.
2) Implementar una función que reciba por parámetro un Zipic y devuelva el tamaño del mismo. Luego:

    a) Escribir un programa de prueba que muestre por la salida estandard varias pruebas.
    b) ¿que pre-condición tiene la función implementada?
    c) Enuncie la post-condición de la función.
    d) Luego de enunciar la post-condición verifique que el nombre de la función y el nombre de los parametros refleje la misma, si no es así modifique los nombres.
    e) ¿Qué validaciones realiza su función o si no realiza validaciones cuales podría realizar?. ¿Por qué pensó en esas validaciones?
3) Implementar una función que reciba por parámetro un Zipic y devuelva un número entero que represente la sumatoria de un Zipic. Luego:
    a) Escribir un programa de prueba que muestre por la salida estandard varias pruebas.
    b) ¿que pre-condición tiene la función implementada?
    c) Enuncie la post-condición de la función.
    d) Luego de enunciar la post-condición verifique que el nombre de la función y el nombre de los parametros refleje la misma, si no es así modifique los nombres.
    e) ¿Qué validaciones realiza su función o si no realiza validaciones cuales podría realizar?. ¿Por qué pensó en esas validaciones?
4) Implementar una función que reciba por parámetro dos Zipic y realice la suma de los mismos . Luego
    a) Escribir un programa de prueba que muestre por la salida estandard varias pruebas.
    b) ¿qué pre-condición tiene esta función?
    c) Luego de enunciar la post-condición verifique que el nombre de la función y el nombre de los parametros refleje la misma, si no es así modifique los nombres.
    d) ¿Qué validaciones realiza su función o si no realiza validaciones cuales podría realizar?. ¿Por qué pensó en esas validaciones?
5) Implementar una función que reciba por parámetro dos Zipic y realice la multiplicación de los mismos . Luego
    a) Escribir un programa de prueba que muestre por la salida estandard varias pruebas.
    b) ¿qué pre-condición tiene esta función?
    c) Luego de enunciar la post-condición verifique que el nombre de la función y el nombre de los parámetros refleje la misma, si no es así modifique los nombres.
    d) ¿Qué validaciones realiza su función o si no realiza validaciones cuales podría realizar? ¿Por qué pensó en esas validaciones?

Implementaciones - Proceso Bernoulli

Se quiere implementar un tipo de dato para representar un Proceso Bernoulli. A continuación se dan definiciones básicas de probabilidad y estadistica, luego se define que es un proceso bernoulli y finalmente se presenta un problema de dominio que se quiere resolver con su implementación.

Experimento Aleatorio (random experiment): Es una acción o proceso el cual esta principalmente caracterizado por estos dos puntos:

- De esa acción o proceso se pueden obtener distintos resultados
- No se conoce a priori el resultado de ese experimento

Al conjunto de resultados posibles se lo conoce cómo **espacio muestral** (sample space).

Suceso o Evento (simple event): Es un subconjunto del espacio muestral. En general me interesa la **probabilidad** de que suceda ese evento. La probailidad expresa un grado de certeza de que ocurra un suceso. Es un número mayor igual que cero (cuando un suceso tiene valor de probabilidad cero es un suceso imposible) y menor e igual que 1.

Veamos algunos ejemplos:

Ejemplo 1: Quiero arrojar un dado y me interesa saber si sale 4 o 5 porque le aposte mucha plata a que sale el número 4 o el número 5 a un amigo.

- El **experimento aleatorio** es: Arrojar un dado una sola vez
- El **espacio muestral** es: {1, 2, 3, 4, 5, 6}. Que son los resultados posibles del experimento aleatorio
- El suceso que me interesa es eobtener el numero 4 o 5 llamemoslo A. Es decir el subconjunto {4,5}. La probabilidad de obtener un 4 es ⅙. La probabilida de Obtener un 5 es ⅙. Luego la probabilidad de obtener un 4 o un 5 es la suma de las probabilidades => P(A) = ⅙ + ⅙ = ⅓

Ejemplo 2: Quiero arrojar 2 veces un dado y me intersa saber si saco primero un 4 y luego saco un 1.

- El **experimento aleatorio** es: Arrojar un dado dos veces
- El espacio muestral es: { (1,1); (1,2); (1,3);(1,4);(1,5);(1,6);(2,1); (2,2); (2,3);(2,4);(2,5);(2,6);(3,1);(3,2); (3,3);(3,4);(3,5);(3,6);(4,1); (4,2); (4,3);(4,4);(4,5);(4,6);(5,1); (5,2); (5,3);(5,4);(5,5);(5,6);(6,1); (6,2); (6,3);(6,4);(6,5);(6,6)}
- El suceso es B = {(4,1)}. Que significa que saco primero un 4 y luego un 1. Notar que en este experimiento aleatorio importa el orden (si no importara el orden el suceso seria {(1,4);(4,1)}). Luego P(B)= 1/36

Ejemplo 3: Quiero ir a la para de colectivo X (donde paran los colectivos A,B,C,D,E,F y G) a las 5 de la tarde del lunes a tomarme el primero colectivo y me interesa saber si me tomo el A, el E o el F

- **El experimento aleatorio es:** tomar el primer colectivo un lunes a las 5 de la tarde en la parada X.
- **El espacio muestral es:** {A,B,C,D,E,F,G}. Que pasa si dos colectivos llegan al mismo tiempo y para uno delante de otro? Cual seria el primero, el que este mas adelante? Vemos que los espacios muestrales son subjetivos.
- El suceso es {A, E, F}. Cual es la probabilidad? En los ejemplos anteriores eran casos en los cuales se consideraron equiprobables (que al tirar un dado salga 1, 2, 3,4,5 o 6 ademas de impredecible es equiprobable cada uno de esos sucesos). En este caso no, se necesita realizar otro procedimiento para asignar una probabilidad, cómo por ejemplo utilizar una funcion de distribución de probabilidad.

Ejemplo 4: La funcion `int rand();` de la libreria standard (stdlib) me permite obtener un número aleatorio del 0 al número definido por la constante `RAND_MAX`. Esta función tambien garantiza que la aparicion de cada número es equiprobable.

- El experimento aleatorio es: Obtener un número random.
- El espacio muestral es: los número enteros del 0 a `RAND_MAX`
- espacio no definimos ningún suceso. Suponga que se quiere definir el siguiente suceso: Obtener un número random del 2 al 10. Haga la implementación de una función en c++ que cumpla con eso. Cual es la probabilidad de obtener un 3? Son equiprobable los resultados de los experimentos aleatorios de su implementación?

Experimiento de Bernoulli: Es un **experimento** que solo tiene dos resulados. A uno de estos resultados se lo conoce cómo de éxito (con una probabilidad **p**) y el otro de fracaso (con una probabilidad q = 1 - p). En otras palabras es un **experimiento** en donde su espacio muestral tiene dos elementos.
Ejemplo:

- En el caso del ejemplo 1 podemos pensarlo cómo experimento bernoulli y decir que:
  - El espacio muestral {Éxito, Fracaso} donde exito seria sacar 4 o 5 y fracaso es sacar otro numero (1,2,3 o6)
  - **p** =P(Éxito) = ⅓
  - **q** =P(Fracaso) = 1 - P(Éxito) = ⅔

Proceso Bernoulli: Consiste en hacer **n** veces un **experimento de Bernoulli** en donde se cumple que:

- Las condiciones no varian.
- Cada suceso es independiente. Es decir que el resultado de un suceso no modifica las probabilidades de ningun otro suceso.

Es decir que un proceso de Bernoulli tiene las siguientes variables

- n: Es la cantidad de veces que se hace el experimento
- p : Es la probabilidad de exito
- k : Es la cantidad de veces que se obtiene éxito en las n veces

En un proceso Bernoulli normalmente se conoce **p** y se utiliza para obtener respuestas sobre las siguientes variables:

- Cuántos experimentos bernoulli (n) hay que realizar para obtener una determinada cantidad de exitos. Estos se conoconen en probabilidad y estadistica cómo distribución Pascal
- Cuantos éxitos (k) se obtiene luego de realizar una determinada cantidad de experimentos bernoulli. Estos se conocen en probabilidad y estadistica cómo distribucion Binomial.

Consigna

1. Implementar en C++ un tipo de dato que permita resolver la familia de problemas de procesos bernoulli para la distribucion **Binomial**.
2. Resolver el siguiente problema de dominio con su implementación
    - Se sabe que en una región la probabilidad de que ocurra un tornado en un dia es de 0.01. Se quiere saber cuantos tornados consecutivos ocurrieron en los últimos 1000000 (2739.7 años)

Interfaz (o contrato), Implementación y Código Cliente

1. Para el ejercicio de Conectividad realizado en clase se pide hacer
    a. Reestructurar el programa en un solo archivo identificando claramente la interfaz, la implementación y el código cliente.
    b. Reestructurar el programa en 3 archivos (main.cpp, Conectividad.h, Conectividad.cpp) para identificar el código cliente, la interfaz y la implementación del tipo de dato
2. Para el ejercicio de Implementación de Número complejo de esta guia se pide
    a. Crear el archivo NumeroComplejo.h con la interfaz de todas las operaciones
    b. Crear el archivo NumeroComplejo.cpp con la implementación
    c. Implementar la siguiente operación: |(2 +2.5i) * (2 +2.5)| en main.cpp
3. Para los ejercicios Zipic de esta guía se pide
    a. Crear el archivo Zipic.h con la interfaz de todas las operaciones
    b. Crear el archivo Zipic.cpp con la implementación
    c. Crear el archivo main.cpp con una implementación que usted elija y que utilice la interfaz de Zipic
4. Para los ejercicios Cadena de Caracteres de esta guía se pide
    a. Crear el archivo CadenaCaracteres.h con la interfaz de todas las operaciones
    b. Crear el archivo CadenaCaracteres.cpp con la implementación

Crear el archivo main.cpp con una implementación que usted elija y que utilice la interfaz de CadenaCaracteres

Ejercicios de diseño de estructuras de Datos

Para cada ejercicio se pide solamente que defina una estructura de datos que represente el modelo pedido. La estructura de datos puede estar compuesta por una o varias estructuras de datos. Se espera que la estructura resultante sea la más conveniente en términos de:

- Representación del problema: Debe ser clara y simple
- Protección de datos: Debe permitir que todas las instancias posibles sean siempre válidas.

Tener en cuenta que por cada ejercicio pueden haber varias representaciones, por lo tanto puede dar más de una respuesta.
Puede declarar o crear instancias de pruebas de sus estructuras de datos.

1) Definir una estructura de datos que represente cualquier carta de una baraja española.
2) Definir una estructura de datos que represente una baraja de cartas españolas.
3) Definir una estructura de datos que represente un dia del almanaque: Dia (numero y nombre de dia de la semana), Mes, Año.
4) Definir una estructura de datos que represente el almanaque de un determinado año.
5) Definir una estructura de datos que represente la hora como un reloj que marca las horas, minutos y segundos.
6) Definir una estructura de datos que permita representar un recordatorio a un día del almanaque. Ejemplo de un recordatorio: El día Jueves 8 de Septiembre.
7) Definir una estructura de datos que permita representar cualquier objeto Json. El objeto Json puede tener cualquier cantidad de atributos y esta solamente limitado a que el valor de sus atributos solo puede ser una cadena de caracteres. Ejemplos de objetos Json que deberia poder representar su estructura de datos
    a) `“{‘Dia’: ‘8’, ‘DiaSemana’: ‘Miercoles’, ‘Mes’: ‘Junio’, ‘Anio’:’2019’}”`
    b) `{“Pelicula”: “La naranja mecanica”, “Director”: “Stanley Kubrick”}`
8) Definir una estructura de datos que permita representar cualquier registro de una cadena de caracteres en formato CSV. El registro puede tener cualquier cantidad d campos. Ejemplos registros CSV
    a) `El padrino,1972,Francis Ford Coppola`

    `Superman,1978,Richard Donner`

Ejercicios de parseo de cadenas de caracteres

Para todos los ejercicios de parseo se pide:

- Interpretar el formato de la cadena de caracteres que se desea parsear, en otras palabras: entender su formato y que es lo que representa
- Definir una estructura de datos que represente la cadena de caracteres que se desea parsear
- Implementar el parseo: Cargar la estructura de datos con la cadena a parsear
- Que el código esté limpio y claro según los criterios trabajados en clase

Parseo de cadenas Json

1) Escribir un programa que parsee y muestre por pantalla un listado el siguiente texto que representa una pelicula en formato Json: `{"Nombre": "La naranja mecánica", "Año":"1971",1978 "Director": "Stanley Kubric"}`
2) Escribir un programa que parsee y muestre por pantalla un listado del siguiente texto que representa un array de películas en formato Json: `[{"Nombre": "La naranja mecánica", "Año":"1971", "Director": "Stanley Kubric"}, {"Nombre": "El padrino", "Año":"1972", "Director": "Francis Ford Coppola"}, {"Nombre": "Superman", "Año":"", "Director": "Richard Donner"}]`. El listado debería estar de la siguiente forma:

```
Pelicula 1
La naranja mecanica
1971
Stanley Kubric

Pelicula 2
El padrino
1972
Francis Ford Coppola

Pelicula 3
Superman
1978
Richard Donner
```

3) Una cadena Json puede representar objetos de transferencia de datos (su abreviatura en ingles es DTO's). En los ejercicios anteriores el DTO era una pelicula que tenía los atributos: Nombre, Año y Director. Se pide implementar un programa que parsee cualquier array de DTO's y los liste por pantalla. Por ejemplo el siguiente array de DTOS `[{"Nombre": "Fabian", "Email" : "fabi@gmail.com", "DNI":"30234174"},{"Nombre": "Superman", "Año":"1978", "Director": "Richard Donner"}, {"Zipic": "1 3 2 -1", "Tamanio": "3"}]`

Parseo de elementos XML

1) Escribir un programa que parsee y muestre por pantalla el siguiente texto que representa una pelicula en formato de elemento XML: `<pelicula nombre="La naranja mecanica" anio="1971" director ="Stanley Kubric"/>`
2) Escribir un programa que parsee y muestre por pantalla el listado del siguiente texto que representa un array de peliculas en formato de elemento xml: `<peliculas><pelicula nombre="La naranja mecanica" anio="1971" director ="Stanley Kubric"/><pelicula nombre="El padrino" anio="1972" director ="Francis Ford`

```
Coppola”/><pelicula nombre=”Superman” anio=”1978” director =”Stanley Kubric”/></peliculas>
```

Parseo de campos en formato CSV

1) Podemos representar las peliculas en formato CSV con la siguiente cadena

```
La naranja mecanica,1971,Stanley Kubric
El padrino,1972,Francis Ford Coppola
Superman,1978,Richard Donner
```

En esta cadena los campos estan separados por coma y los registros por nueva linea. Escribir un programa que parsee cualquier formato CSV y lo muestre por pantalla del siguiente modo:

```
Registro 1 con 3 campos
Campo 1: La naranja mecanica
Campo 2: 1971
Campo 3: Stanley Kubric

Registro 2 con 3 campos
Campo 1: El padrino
Campo 2: 1972
Campo 3: Francis Ford Coppola

Registro 3 con 3 campos
Campo 1: Superman
Campo 2: 1978
Campo 3: Richard Donner
```

Ejercicios de escritura de pruebas unitarias

Por cada ejercicio se pide:

- Pensar varios escenarios de pruebas unitarias sobre la función dada.
- Implementar varias de las pruebas unitarias de manera tal que se asegure que la función se comporta de la manera esperada (cumple con su responsabilidad). Escriba al menos 5 pruebas.
- La implementación de cada prueba unitaria debe
    - Ser una función que realiza una única prueba unitaria (es decir que hay una función por cada prueba unitaria)
    - Cada función debe tener el el código claro y simple en donde se vea
        - Como se inicializa el contexto de la prueba
        - Cual es la prueba, es decir ejecutar la funcion con los parametros inicializados en el paso anterior (el nombre de la función debe ser acorde a la prueba y el resultado esperado)
        - Como se verifica la prueba y se muestra el resultado obtenido (el resultado solo debe decir si pasó la prueba o no)
- Si las pruebas encuentran un error en la función debe corregir la función de manera tal que pase todas las pruebas

1) Escribir las pruebas unitarias para la siguiente función (link de ayuda):

```
bool SonIguales(char cadena1[],char cadena2[])
{
  int indice=0;
  bool sonIguales=true;
  bool finCadena = cadena1[0]=='\0' || cadena2[0]=='\0';
  while(sonIguales && !finCadena)
  {
          sonIguales = cadena1[indice] == cadena2[indice];
          ++indice;
          finCadena = cadena1[indice]=='\0' || cadena2[indice]=='\0';
  }

  return sonIguales && cadena1[indice]=='\0' && cadena2[indice]=='\0';
}
```

2) Escribir las pruebas unitarias para la siguiente función:

```
bool EsZipic(int zipic[])
{
  bool tieneMarcaFin = false;
  bool todosSonEnterosPositivos = true;
  bool esZipic = false;
  int indiceZipic = 0;
  while(indiceZipic<Tope && todosSonEnterosPositivos && !tieneMarcaFin)
  {
          if(zipic[indiceZipic]==-1)
          {
                  tieneMarcaFin = true;
          }
          else
          {
                  todosSonEnterosPositivos = zipic[indiceZipic] > 0;
          }
          ++indiceZipic;
  }
```

```
  esZipic = tieneMarcaFin && todosSonEnterosPositivos;
  return esZipic;
}
```

3) La siguiente función no se encuentra implementada aún. Solamente se pide que implemente las pruebas unitarias (es una buena práctica de programación implementar las pruebas unitarias antes de implementar una funcionalidad)

```
const int TopeCantidadCampos;
const int TopeTamanioCampo;

struct CamposSeparados{
  char campos[TopeCantidadCampos][TopeTamanioCampo];
  int cantidadCampos;
}


// Precondicion: @campos y @separadorCampos son cadenas de caracteres
// Postcondicion: Obtiene un CamposSeparados con las siguientes características
//  - CamposSeparados::campos->tiene todos los campos separados por @separadorCampos en @campos
// - CamposSeparados::cantidadCampos-> contiene la cantidad de campos encontrados
// Ejemplo:
// Cuando se invoca de la siguiente manera: SepararCampos("hola||que||tal", "||")
// se obtiene las siguientes características:
// - CamposSeparados::cantidadCampos es igual a 3
// - CamposSeparados::campos es igual a la matriz:
// 'h' 'o' 'l' 'a' '\0'
// 'q' 'u' 'e' '\0'
// 't' 'a' 'l' '\0'
CamposSeparados SepararCampos(char campos[], char separadorCampos[]);
```