INFORMATICA II	SUBTEMA	TEMA #	PARCIAL	26/06/2021
	C++/IPC	5	# 1	

Estimado alumno,

A partir de la siguiente página encontrará el enunciado de su examen. Le recordamos que ya no puede cambiarlo.

Ud. deberá resolver el ejercicio en su computadora. Luego, deberá subir los archivos realizados comprimidos (preferentemente zip). Si lo desea, puede agregar un documento de texto con explicaciones que quiera hacerle llegar a su profesor.

Cada aproximadamente una hora debe actualizar su repositorio GIT con lo que tenga hasta ese momento resuelto.

Le recordamos:

	Fecha	Tópico	Examen disponible durante	Tiempo para hacerlo desde bajada	Modalidad	Entregables
1	**Sáb 26/6 desde las 8am hasta Dom27 – 19hs.**	C++/IPCs	35hs.	4hs. (*)	Ejercicio en compilador (C++)	Archivo .zip (ver NOTA 1)
3	**Sáb 17/7 desde las 8am hasta Dom18 – 19hs.**	MdE	35hs.	4hs. (*)	Ejercicio en uModel Factory	Archivo .zip (ver NOTA 2)

(*) La duración nominal del examen se cuenta desde el momento en que el alumno comienza a realizar el examen dentro del Aula Virtual (AV). La cuenta del tiempo la lleva la plataforma Moodle automáticamente. Si bien es cierto habrá 35hs a partir del horario de comienzo para comenzar la evaluación, y que las evaluaciones en sí pueden realizarse en 4 horas, debe tenerse presente que, a las 18:59:59 del día domingo quedará bloqueada la subida del archivo final. En criollo: Si va a tomar el examen de MdE a las 18hs del domingo 18 de julio, solo tendrá una hora para entregarlo.

NOTA1: con los fuentes del proyecto/ejercicio + archivo de texto con aclaraciones, de considerarlo necesario el alumno.

NOTA2: con los fuentes del proyecto + el archivo .umf correspondiente a la MdE realizada en uModel Factory + archivo de texto con aclaraciones, de considerarlo necesario el alumno.

Cátedra de Info2

1.-Se requiere desarrollar una clase, la cual llamaremos ***sText***, para el manejo de nombres de marcas y modelos de productos, para lo cual se definió utilizar como soporte la clase ***std::string*** que provee el lenguaje a través de una composición.

Un objeto de esta clase, debe tener las siguientes propiedades.

- El texto a guardar, debe poseer uno o dos palabras, no más.
- Los únicos símbolos que se aceptan son los alfanuméricos, incluyendo el espacio como separador entre palabras (es decir que los símbolos como +-*/$%#@!.,<> y demás deben ser rechazados)
- Independientemente de cómo se cargue el dato, siempre deberá ser utilizado con la primera letra de cada palabra en mayúscula y las demás en minúscula (se recomienda almacenarlo directamente de esta manera). Cabe aclarar los datos numéricos se mantendrán invariables.
- Una vez que el parámetro es correctamente almacenado, no debe poder ser modificado.
- Debe contar con un método que permita saber si el objeto ha sido correctamente cargado o no.
- Debe cumplir con el principio de encapsulamiento. Esto es, no se debería poder modificar ciertos atributos desde fuera del objeto.

Debe contener al menos los siguientes constructores y métodos

- Constructor sin parámetros
- Constructor con parámetros
- Constructor de copia y/o destructor, solo si fuese necesario
- setNombre, para configurar el nombre, solo si el mismo no ha sido ya configurado
- getNombre, que retornará un string con el nombre configurado (si ya ha sido configurado) o NULL en caso contrario.
- operator== , que permitirá comparar si dos elementos sText son iguales o no. Debe permitir comparar un elemento sText con un string y con un char *.
- operator!= , debe implementarse como la negación de operator==.
- Un método que permita saber si el objeto ya ha sido configurado.
- operator<<,entre una clase ostream y una sText, que debe imprimir el valor configurado en sText, o el texto “SIN TEXTO” en caso de no haber sido debidamente configurado.

Queda a criterio del alumno definir:

- Cómo actuará la clase si se intenta configurar con un string de más de 2 palabras o si el texto posee caracteres inválidos, entre otros posibles errores de configuración.
- Los campos internos de la clase.

Nota: *Si no maneja strings, puede implementarlo con char**

2.- Como parte de un sistema de gestión de una empresa, debe crear una clase base para el manejo de productos.

Los parámetros que debe poseer esta clase son:

- Marca - tipo sText
- Modelo - tipo sText
- Código – número entero de 5 cifras que no empiece con cero
- Color – número entero de 0 a `0x00ffffff`.

Además, debe contar con un indicador que permita saber si el objeto ha sido bien creado o no, esto es, posee un color y código válidos, además de contar con una marca y modelo válidos.

La clase desarrollada debe cumplir con los conceptos de encapsulamiento y protección de sus datos. Es decir que debe cuidar que no se pueda ingresar libremente a sus variables miembro.

Para esta clase, se pide:

Desarrollar los siguientes constructores:

- Sin parámetros
- Con todos los parámetros

Desarrollar los siguientes métodos

- operator== , que permita comparar si dos objetos de la clase son exactamente iguales. Este debe retornar true, solo si ambos objetos han sido correctamente inicializados y poseen sus parámetros iguales.
- Un método que permita saber si el objeto posee todos sus campos (esto es Marca, Modelo, Codigo y Color) válidos, es decir que han sido expresa y correctamente configurados.
- Un método que permita imprimir, al menos en pantalla, los parámetros del objeto, aprovechando el código que ya se encuentra desarrollado.

Realizar además una aplicación que permita verificar las características solicitadas de la clase.

IPCs:

Dada la siguiente clase, realizada para implementar una Shared Memory:

```
class IPC
{

protected:
    key_t llave;
    bool borrar;
    bool generateKey(char * p = nullptr , char a = 0);

public:

    static const int DISCONNECTED = -1;

    IPC(char * p = nullptr , char a = 0 , bool destroy = false);
    key_t   getKey  (void) const;
    void    borrarAlFinal   (bool);
};
```

```
class ShMem : public IPC
{
private:
    int id_shmem, id_sem;
    char *buffer;
    int size;

    void bloquearShMem();
    void desbloquearShMem();

public:
    ShMem   (char * p = nullptr , char a = 0, int tam = TAM , bool destroy = false);
    ShMem   (const ShMem &);

    bool    conectar( char * , char );
    bool    conectar( char *, char , int );

    int     getID   (void) const;
    int     getSize (void) const;

    bool    escribirMensaje ( void * msj , int cant , int pos = 0 );
    int     leerMensaje ( void * buf , int pos = 0 ,  int cant = TAM );

    ~ShMem();
};
```

Se pide:

a. Desarrollar el método ***bool conectar (char *, char , int)***, que recibe como parámetros:
   - Un puntero a char, para albergar una cadena de caracteres que hace referencia a un nodo del filesystem.
   - Una variable de tipo char.
   - Un entero, que indica el tamaño de la shared memory a generar.

   Dicho método deberá verificar que la shared memory no se encuentre creada (en cuyo caso ID será distinto a ***IPC::DISCONNECTED***), y de comprobarse este punto deberá generar una llave a partir de los 2 primeros parámetros recibidos y crear y conectarse a una shared memory a partir de esta llave. Adicionalmente, se deberá crear y conectarse a un semáforo para controlar la escritura y lectura de la misma, utilizando la misma llave. Las variables privadas ***llave, id_shmem, buffer*** y ***id_sem*** deberán quedar cargadas con los valores devueltos por dichas funciones. De producirse algún error, la función devolverá *false*, y id_shmem quedará cargado con el valor ***IPC::DISCONNECTED***). En caso contrario el método devolverá *true*..

b. Realizar el destructor de la clase ShMem, que deberá eliminar los IPCs en caso de que la variable booleana ***borrar*** sea true.