INFORMATICA II	SUBTEMA	TEMA #	PARCIAL	26/06/2021
	C++/IPC	10	# 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)

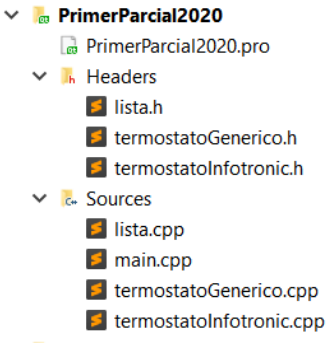
(*) La duración nominal del examen se cuenta desde el momento en que el alumno comienza a realizar el examen dentro del Aula Virtual (AV). La cuenta del tiempo la lleva la plataforma Moodle automáticamente. Si bien es cierto habrá 35hs a partir del horario de comienzo para comenzar la evaluación, y que 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

C++:



Usted cuenta con un proyecto de QT denominado "PrimerParcial2021" (que se le provee con el examen) donde ya está implementada la función main (**que no debe modificar**) y el esbozo de tres clases que deberá modificar/completar.

Clase Lista: lista simplemente enlazada que se utilizará para el almacenamiento temporal de muestras de temperaturas.

Se debe implementar:

- *el "constructor" de la clase que inicializa la lista como vacía;*
- *el "destructor", encargado de liberar la memoria utilizada.*
- *el método "agregar", encargado de insertar al final de la lista la temperatura que recibe por parámetro.*
- *el método "obtenerPromedio", encargado de calcular el promedio de temperatura y liberar la memoria que haya sido utilizada hasta el momento.*

Nota: El promedio es la suma de todas las temperaturas previamente guardadas en los nodos (mediante el método agregar) dividido la cantidad de nodos. Si la lista está vacía, el promedio es cero.

Clase termostatoGenerico: La explicación de esta clase surgirá de los comentarios en los archivos fuente.

Se debe implementar:

- *el "constructor" de la clase, encargado de inicializar el modelo (o nombre) del termostato e inicializará el objeto m_temperatura, que será la lista donde posteriormente se almacenarán las muestras de temperaturas.*
- *el "destructor", encargado de liberar la memoria solicitada.*
- *el operarador sobrecarado "+=", que deberá insertar en la lista la temperatura pasada como parámetro.*

Clase termostatoInfotronic: Hereda de la clase termostatoGenerico, y ya se encuentra implementada(ver fuentes)

El objetivo del ejercicio es lograr que por consola se vea lo siguiente:

```
A Primer parcial - C++
B Nuevo termostato, modelo: Infotronic
C Numero de serie: 1
D Temperartura promedio = 33.41
E Temperartura promedio = 0
F Temperartura promedio = 33.5
G Cantidad de termostatos producidos hasta el momento: 1
H Nuevo termostato, modelo: Infotronic
I Numero de serie: 2
J Cantidad de termostatos producidos hasta el momento: 2
  Press <RETURN> to close this window...
```

A modo de referencia se indica en que línea del main se imprime cada salida por consola:

```cpp
int main()
{
    std::cout << "Primer parcial - C++" << std::endl;

    TermostatoInfotronic termostato;   // Se crea una instancia de la clase TermostatoInfotronic
    std::cout << "Numero de serie: " << termostato.numeroDeSerie() << std::endl;

    // Se agregan muestras de temperatura mediante el operador sobrecargado +=
    termostato += 33.22;
    termostato += 33.74;
    termostato += 33.70;
    termostato += 33.39;
    termostato += 33.21;
    termostato += 33.12;
    termostato += 33.56;
    termostato += 33.34;
    std::cout << termostato;    // Imprime el modelo y el promedio de temperatuaras previamente agregadas
    std::cout << termostato;    // Idem. El promedio en este punto debe dar cero (debido a que no hay muestras de temperatura)

    // Se agreagan una segunda ronda de muestras de temperaturas
    termostato += 33.00;
    termostato += 33.50;
    termostato += 34.00;
    std::cout << termostato;    // Imprime el modelo y el promedio de temperatuaras previamente agregadas

    // Imprime la cantidad de termostatos infotronic "producidos"
    std::cout << "Cantidad de termostatos producidos hasta el momento: " << TermostatoInfotronic::cantidadProducida() << std::endl;

    const TermostatoInfotronic termostato2;   // Se crea una instancia de la clase TermostatoInfotronic
    std::cout << "Numero de serie: " << termostato2.numeroDeSerie() << std::endl;
    std::cout << "Cantidad de termostatos producidos hasta el momento: " << TermostatoInfotronic::cantidadProducida() << std::endl;

    return 0;
}
```

Importante:
Se recomienda realizar el ejercicio por etapas, para ello usted puede comenzar "comentando" todo el main y las clases "termostatoGenerico" y "termostatoInfotronic" y concentrarse en la resolución de la clase lista. Una vez que no tenga errores, puede pasar a resolver "termostatoGenerico". Vaya "descomentando" parcialmente el main a medida que avanza en la resolución.
Ejemplo, puede dejar solo la línea 9 donde se instancia el objeto termostato. Debe lograr que ejecute sin errores, luego puede "descomentar" la siguiente línea, donde se hace uso del método numeroDeSerie()… y así siguiendo hasta terminar.
RECUERDE QUE NO PUEDE MODIFICAR NADA EN EL ARCHIVO MAIN (a excepción de la siguiente nota).

Nota: el objeto termostato2 es del tipo const. Si esta característica le trae problemas en la compilación y no logra resolverlo, remueva la palabra const (esto baja la calificación, pero le permitirá avanzar con la resolución del ejercicio).

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 generateKey (char *, char)***, 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.

   Dicho método deberá generar una llave a partir de los 2 primeros parámetros recibidos, y cargar la variable llave con el resultado de la operación. De producirse algún error, la función devolverá *false*, y la variable llave quedará cargada con (key_t)-1. En caso contrario el método devolverá *true*..

b. Realizar el método ***bool escribirMensaje (void *, int , int)***, cuyos parámetros son:
   - Un puntero donde se encuentra el mensaje que se desea escribir.
   - Un entero que indica el tamaño del mensaje.
   - Un entero que indica la posición de la memoria compartida a partir de la cual se escribirá el mensaje.

   El método deberá escribir el mensaje recibido en la memoria, asegurándose que ningún otro proceso se encuentre escribiendo o leyendo la misma en ese momento. Para la realización de este método utilizar los métodos ya implementados cuando sea necesario, descriptos en la definición de la clase.