INFORMATICA II	SUBTEMA	TEMA #	PARCIAL	26/06/2021
	C++/IPC	12	# 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++:

Se cuenta con una clase Punto y otra Polígono

```
class Punto
{
public:
    Punto(int x = 0, int y = 0);
    int getX() const;
    int getY() const;
    void setXY(int x, int y);

private:
    int m_x, m_y;
};
```

```
class Poligono
{
public:
    Poligono(int numeroDeLados = 3);
    ~Poligono();
    const Punto &getVertice(int numeroDeVertice) const;
    Poligono& operator <<(const Punto &);
    Poligono& operator =(const Poligono &);

private:
    static int m_cantidadDePoligonos;
    Punto *m_vertices;
    int m_verticesIncorporados;
}
```

Como puede observarse en la clase Polígono, existe lo siguiente:

- Un constructor que indica cuantos vértices tiene un objeto creado.
- Un destructor de la clase.
- Un método para “leer” alguno de los vértices del polígono.
- El operador << sobrecargado que permite agregar los vértices al objeto Polígono.
- El operador = también sobrecargado para la copia.

En cuanto a los miembros son los siguientes:

- Uno estático “m_cantidadDePoligonos” que debe contabilizar la cantidad de objetos Polígono creados,
- Un puntero “m_vertices” que deberá apuntar a un array de objetos Punto (se debe pedir memoria en forma dinámica para tantos Puntos como lados tenga el Polígono),
- Un miembro m_verticesIncorporados que permite contabilizar cuantos vértices se han “incorporado” al mediante el operador <<.

Ejemplo:

```
Poligono triangulo(3);
Punto vertice1(0, 10), vertice2(10,15), vertice3(8,7);
triangulo << vertice1;
triangulo << vertice2;
triangulo << vertice3;
```

Se pide:

1. Implementar la clase Punto tal como se encuentra diseñada.
2. Implementar el constructor y el destructor de la clase Polígono, el método operador <<, el método operador = y el método getVertice, de manera que la clase se comporte de acuerdo a la descripción de la misma.
3. Implemente el método ***ostream& operator << (ostream &o, Poligono &);*** Este método tiene que imprimir en pantalla todos los puntos del polígono.
4. Incorpore a la clase Punto un método que permita comparar 2 puntos mediante el operador ==, y luego utilícelo en la clase Polígono para eliminar uno de los puntos del Polígono, utilizando las siguientes instrucciones:

```
Poligono figura(4);
Punto vertice1(0, 10), vertice2(10,15), vertice3(8,7), vertice4(3,6);
figura << vertice1;
figura << vertice2;
figura << vertice3;
figura << vertice4;
figura -= vertice3; //Elimina un vértice de la figura convirtiendola en un triangulo
```

Nota: Si el número de lados es menor a tres, ningún método debe realizar ninguna acción.

IPCs:

1. Dada la siguiente clase, realizada para implementar un Message Queue:

```
class IPC
{

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

public:

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

```
class MsgQueue : public IPC
{
private:
    struct mymsgbuf{
        long mtype;
        char mtext[MAX];
    };

    int id;

public:

    static const int DISCONNECTED = -1;

    MsgQueue(char * p = nullptr , char a = 0 , bool destroy = false);
    MsgQueue(const MsgQueue & a);
    bool    conectar( char * , char );
    int     getID   (void) const;
    bool    enviarMsj  ( void * , int , long typ = 1) const;
    int     recibirMsj ( void * , int , long typ = 0) const;
     ~MsgQueue();

};
```

Se pide:

a. Desarrollar el método ***bool conectar (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 verificar que la cola no se encuentre conectada (en cuyo caso ID será distinto a ***MsgQueue::DISCONNECTED***), y de comprobarse este punto deberá generar una llave a partir de los parámetros recibidos y conectarse a una message queue a partir de esta llave. Las variables privadas ***llave*** y ***id*** deberán quedar cargadas con los valores devueltos por dichas funciones. De producirse algún error, la función devolverá *false*, y id quedará cargado con el valor ***MsgQueue::DISCONNECTED***). En caso contrario el método devolverá *true*..

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