Tel-102 Par 200/201
2º Semestre 2020
Práctico 3
27/10

Instrucciones

- Lea con detenimiento cada una de las actividades a realizar durante la experiencia.
- Cree un archivo con extensión **.cpp** con lo desarrollado. El nombre del archivo debe tener el siguiente formato:
  TEL102_P3_Nombre_Apellido.cpp, sin tíldes. (Ej. TEL102_P3_Mauricio_Araya.cpp)
- Enviar el archivo a través de la página de aula del ramo, sección "Práctico 2" hasta las 23:59:59 del día de mañana 27/10/2020 hora local continental de Chile (UTC-3).
- Trate de utilizar herramientas conocidas o aprendidas en clases. **No copie literalmente de recursos online**.
- Sea riguroso con las instrucciones de desarrollo.
- Comente adecuadamente el programa, describiendo lo que hace.
- ¡Éxito!

Voto Telemático

El **Tricel++** de la carrera de Ingeniería Civil Telemática de la **UTFSM++**, la universidad más prestigiosa de **PlusPlusCity**, ha decidio incursionar en un sistema de voto telemático debido a la pandemia, es decir, que los votantes podrán ejercer su derecho a sufragio utilizando las redes telemáticas disponibles (i.e., Internet). El **Tricel++** ha decidido utilizar programación orientada a objetos, la cual permitirá hacer pruebas, simulaciones e ir desarrollando el sistema por etapas. Además, el *control de acceso*[1] provisto por los lenguajes orientados a objetos, será una de las barreras de seguridad del sistema (entre otras como encriptación, seguridad en redes, etc.)
En esta primera etapa se requiere implementar los tres elementos centrales de una votación: el voto, el padrón y la urna. El voto es secreto, es solo uno por persona y debe contabilizarse las opciones solo en el momento del escrutinio (¡no antes!). Para evitar dobles votos, se cuenta con un padrón, donde se registra quien ha votado ya. El voto se inserta en una urna, y nadie puede sacar los votos hasta que la urna sea abierta para su escrutinio.
El **Tricel++** ha solicitado sus servicios como programador, principalmente porque el arquitecto de software faltó el día que se le enseñaría *control de acceso*. En consecuencia, se le entrega el siguiente encabezado (header) con la definición de las clases a implementar, pero con todos los modificadores de acceso como `public`.

[1] Público, Protegido, Privado

practico3.h

```
#include <iostream>

class Voto{
        public:
                Voto();

                void setCodigo(int cod);
                void setOpcion(int cod, int opt);

                int leer(int cod);
                int codigo;
                int opcion;
};

class Padron{
        public:
                Padron();

                ~Padron();

                void sign();
                bool checkFirma();
                bool *firma;
};

class Urna{
        public:
                Urna(int cod_mesa, int votantes);

                ~Urna();

                bool votar(int cod_voto,int opcion);
                void escrutar(int cod_mesa);

                bool sellada;
                int codigo;
                Padron *libro_firmas;
                Voto *votos;
                int sufragios;

};
```

La situacion del **Tricel++** es la siguiente: `Cristoforo Milo` se está respostulando a presidente del Centro de Estudiantes de Telemática. Para esta elección cada votante debe elegir entre la `opcion` 1 (Apruebo) o 2 (Rechazo) para emitir su voto, mientras que cualquier otro valor equivale a anular/blanco y se le asigna el valor 0.
Además, al sufragar se le entrega un `codigo` único al voto, para que éste se pueda `leer` solo con

dicho código, sino el voto devolverá -1 indicando que el código es erróneo. Por otro lado, el libro de firmas consiste en un arreglo de la clase Padrón (V/F) y tiene tamaño igual a la cantidad de `votantes` de una Urna. Éste libro indica si un votante ya participo y puso su `firma`. Cada vez que un elector vota, se debe llamar al método `setOpcion` para registrar que el elector con el `codigo` correspondiente (índice del arreglo) ya votó por una `opcion` . En el caso que ese elector ya votase, se debe hacer notar dicha situación y no contabilizar el voto.
La clase `Urna` se inicializa con un `codigo` de mesa secreto para escrutar la Urna sólo cuando termine la votación. Sin este código, los votos no se pueden leer (ver párrafo anterior), es decir la Urna se mantendrá `sellada`. La urna contiene los `votos`, aunque técnicamente contiene un arreglo de objetos del tipo `Voto`. Cuando se llama al método `votar`, se intentará registrar el voto en el libro de registro y se generará un nuevo voto en la lista de votos, siempre y cuando no votase ya. El método retornará falso si el voto no se pudo efectuar y verdadero en otro caso. Para evitar cohecho, el registro solo podrá ser leído solamente mediante el método `escrutar`. Al llamar a este metodo, la Urna deja de estar `sellada`, y se reportan por salida estándar la suma de los votos por opción 1, opción 2 y nulos/blancos (0), siempre y cuando el voto efectuado esté también marcado en el registro de firmas. Recuerde que los codigos de urnas y votos son internos y nadie fuera del codigo debería conocerlos.

1. (5p) Implemente las clases `Voto`, `Registro` y `Urna`, recordando en estas dos últimas liberar la memoria de heap solicitada en sus destructores.

2. (3p) Implemente la función `main` para que solicité por entrada estándar un código secreto para crear la Urna, y luego el ingreso de número de electores y voto de forma repetitiva hasta completar la elección. Para finalizar, se debe escrutar los votos, y entregar los resultados.

3. (2p) Agregue los modificadores private (y/o protected) correspondientes para incrementar la seguridad de su código.