Ayudantía 9
Diseño y Programación Orientado a Objetos

Matías Soto S.

23 de Junio 2022

1. Preguntas Teóricas

1. Para cada caso del lado derecho indique la salida esperada o error de compilación producido. No considere errores por ausencia de archivos de encabezados o similar.

```
class A {
public:
  int x;
  A () {x=3;}
  int * get_px(); //se supuso *
  int & get_rx();
};

int * A::get_px(){
   return &x;
}

int & A::get_rx(){
   return x;
}
```

Caso a)
```
A v;
int *p = v.get_px();     (*p)++;
cout << "v.x=" << v.x << endl;
```

Caso b)
```
A v;
int &r = v.get_px();   r++;
cout <<  "v.x=" << v.x << endl;
```

Caso c)
```
A v;
v.get_rx() = 7;
cout << "v.x=" << v.x << endl;
```

Caso d)
```
A v;
A *p=&v;   v.x--;
cout << "p->get_rx()=" << p->get_rx()<<endl;
```

Figura 1: Pregunta 1 teórica.

2. Para el archivo b.h correspondiente a la declaración de la clase B, proponga una implementación simple para la clase B (b.cpp) que incluya un puntero no nulo para el atributo p. (No agregue nuevos métodos ni constructores)

```
// b.h
class B {
    private:
        const int id;
        static int id_global;
    public:
        B();
        ~B();
        int * p;
        int getId();
}
```

3. Considere el siguiente código:

```
class A{
      int x=0;
};

void showX(A a){
   cout << "El valor de x es: " << a.x << endl;
}
```

```
int main(){
   A p;
   showX(p);
   return 0;
}
```

Figura 2: Pregunta 3.

Modifique el código de la clase A sin cambiar el nivel de acceso de su atributo x para que el código dentro de la función main se ejecute sin errores. Justifique. No considere errores por falta de cabeceras ni similares.

4. Se tiene la siguiente definición:

```
class Persona {
    public: void displayNombre();
}
class Estudiante: protected Persona {
    public: void displayNombre();
}
```

Para la siguiente implementación:

```
Estudiante *student = new Estudiante();
Persona *person;
person = student;
person->displayNombre(); // <-- [1]
```

Mencione y justifiqué qué modificador se debería agregar para que el código en [1] ejecute la implementación de displayNombre() de la clase Estudiante.

2. Pregunta de Desarrollo

Queremos almacenar estudiantes de ingeniería (E_ingenieria) y de posgrado (E_posgrado) en un único vector de punteros a Estudiantes. Los estudiantes de ingeniería tienen una carrera y para los de posgrados se registra su número de publicaciones. Se cuenta con la declaración de la clase Estudiante la cual usted no debe modificar. Se le pide implementar la clase Estudiante (estudiante.cpp) y declarar (.h) e implementar (.cpp) las clases E_ingenieria y E_posgrado de manera que el código main dado arroje como salida:

```
Juan es estudiante de Ing. Civil Tel.
Claudia es estudiante de posgrado con 2 publicaciones.
Claudia ha publicado más.
```

El código base se encuentra disponible en el repositorio