**PROGRAMACIÓN APLICADA A LA AUTOMATIZACIÓN**

**PRÁCTICA # 6 Desarrollo de aplicaciones en C++ empleando composición de clases.**

**CAPÍTULO DEL CURSO:** Programación orientada a objetos y estructuras de datos aplicadas a automatización.

**TEMA DE LA ACTIVIDAD:** Creación de programas que usen clases, clases compuestas y objetos.

**NOTA**

**OBJETIVOS DE APRENDIZAJE:**

* Elaborar algoritmos que empleen composición de clases para la creación de programas en C++ que resuelvan problemas prácticos.

**DURACIÓN:** 120 minutos

**MATERIALES Y HERRAMIENTAS:**

* Computadora con Qt Creator correctamente instalado o algún otro editor de C o de texto.

**INTRODUCCIÓN**

El lenguaje C++ debe su popularidad al uso de la Programación Orientada a Objetos. La POO, utilizando el concepto de encapsulamiento, es un paradigma de programación que busca acercar al programador al problema y su solución, y alejarlo de los detalles de la computadora y el lenguaje. Además, tiene la gran ventaja de la reutilización del código previamente diseñado lo que hace más eficiente el trabajo. Dentro de la POO, por ahora, se pueden destacar elementos como las “Clases”, que incluyen datos y también funciones miembros, constructores y destructor. Una manera de reutilización del código previamente diseñado en POO es la composición de clases, que permite construir nuevas clases utilizando otras como sus partes. En esta parte, se realizará un ejercicio bastante completo de uso de clases y su composición. Se debe emplear como bibliografía la conferencia VI o sesión de clase 6 y la propia bibliografía allí propuesta.

**DESCRIPCIÓN DE LA PRÁCTICA:** El estudiante debe implementar la solución del ejercicio, agregarla al informe y subirlo antes de la fecha y hora prevista.

**PROCEDIMIENTO:**

1. Realice el programa en lenguaje C++ necesario para resolver la pregunta o problema que se encuentra posteriormente en este documento.
2. Incorpore dicho programa al informe de esta prepráctica en formato texto.
3. Suba el informe a AulaVirtual antes del comienzo de la práctica.

**Ejercicio 1**

Dada la siguiente definición de clase:

**class** **Screen**

{

**private:**

**int** width, height; //atributos

**public**:

Screen(int, int); //Constructor

**bool** setResolution(int, int); **//**Este metodo permite crear variables que devuelve True o False.

**void** print() **const** {cout << width <<"x"<< height << " pixels" << endl;}; //metodo inline

};

1. Implemente los métodos u operaciones de la clase que no están definidos, conociendo que la pantalla solo puede tener tres posibles resoluciones: **340 X 600, 800 X 600 y 1024 X 768**.
2. Defina una clase **Computer** que está formada por los siguientes miembros:
   1. Una variable de tipo Screen.
   2. Un número entero que represente el tamaño de su disco duro.
   3. Un número entero que represente el tamaño de memoria.
   4. Un número real que represente la velocidad del procesador.
   5. Un dato de tipo de carácter dinámico que represente el nombre de la computadora
   6. Defina un constructor por omisión para esta clase donde la memoria sea de 2GBytes, el disco de 500 GBytes, la velocidad 1,77 GHz y el nombre Computadora01
   7. Defina un método para imprimir la información en pantalla.

Desarrollar en el **programa principal:**

* Una consola para administrar una Bodega, que tenga 2 opciones: registrar un objeto de clase Computer en el inventario y mostrar el inventario.
* Cree una variable de tipo arreglo dinámico de punteros a computadoras (no es un arreglo dinámico de computadoras sino de punteros a computadoras, debe usar **\*\***) con capacidad para 5 elementos.
* Cree una variable tipo entero cantidad para conocer cuántos elementos se van registrando en el arreglo

**Preguntas**

**¿Cuál es la diferencia entre la capacidad de elementos de un arreglo y la cantidad de elementos almacenados en un arreglo?**

**¿Por qué son necesarios los métodos Get y Set?**

**¿Para qué se utiliza la validación en constructores y métodos Set?**

**Si se realiza una composición de clases, ¿Se debe crear nuevos métodos de modificación y acceso para los atributos de las clases primarias?**

**IMPLEMENTACIONES Y RESULTADOS**

**Ejercicio #1:**

[Texto del código]

**CONCLUSIONES Y RECOMENDACIONES**

**BIBLIOGRAFIA**

[1]. Deitel and Deitel. Como programar en C/C++. Segunda edición o superior.

[2]. García de Jalón, J.; y otros. Aprenda C++ como si estuviera en primero. Universidad de Navarra.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **INICIAL** | **EN DESARROLLO** | **DESARROLLADO** | **EXCELENTE** |

**Firma del Profesor**