**PROGRAMACIÓN APLICADA A LA AUTOMATIZACIÓN**

**PRÁCTICA # 8 Desarrollo de aplicaciones en C++ utilizando herencia y polimorfismo.**

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

**TEMA DE LA ACTIVIDAD:** Creación de programas que apliquen el polimorfismo.

**NOTA**

**OBJETIVOS DE APRENDIZAJE:**

* Elaborar algoritmos que empleen herencia y polimorfismo 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**

La programación orientada a objetos presenta tres paradigmas fundamentales: el encapsulamiento, la herencia y el polimorfismo. Al aplicar dichos tres paradigmas en una aplicación práctica se puede ejemplificar un modelado muy completo de la solución de un problema. Además, de esta forma se perfecciona la reutilización del código previamente diseñado. Al utilizar el polimorfismo se pueden manejar colecciones de objetos de diferentes tipos o Clases en un mismo arreglo o listado como si fueran de un mismo tipo o Clase mediante punteros a clase base, siempre que todos los objetos pertenezcan a la misma jerarquía de clases (tengan la misma clase base). En esta parte, se realizará un ejercicio bastante completo de uso de clases, herencia y polimorfismo. Se debe emplear como bibliografía la conferencia VIII o sesión de clase 8 y la propia bibliografía allí propuesta.

**DESCRIPCIÓN DE LA PREPRÁCTICA:** El estudiante debe probar su correcta ejecución, si fuese necesario corregir los errores, y llegar a conclusiones sobre esta implementación. Luego debe realizar la modificación y probarla.

**PROCEDIMIENTO:**

1. Crear una carpeta en el escritorio que tenga de nombre “practica\_8” donde guardará sus programas y luego podrá llevárselos con usted. (Si se hace en su PC propia puede obviar este paso, pero debe saber dónde se guardarán sus proyectos.)
2. Abra Qt Creator.
3. Implemente y pruebe la solución de la pregunta editando su programa en un fichero .cpp. Copie el código del programa al informe de esta práctica (como texto) y la(s) ventana(s) de visualización dónde se vea su ejecución de forma válida (como imagen).
4. Informe al profesor cuando haya concluido.
5. Repita los mismos pasos para la modificación.
6. Responda las preguntas.
7. Describa las conclusiones y recomendaciones de la práctica.
8. Suba el informe a Aula Virtual.

**EJERCICIO:**

Se desea automatizar la gestión de una empresa constructora de edificaciones que se dedica a la fabricación de diferentes obras. Estas son: **casas**, **hoteles** y **centros comerciales** **(malls)**. Cada obra consta de un **nombre**, la cantidad de **metros cuadrados** y un **precio** por metro cuadrado. Además, se conoce que el **tiempo de fabricación** de cada obra es de: 1 mes para una casa, 2 meses por piso para un hotel (cada hotel tiene un **número de pisos**) y 1 mes por departamento para un centro comercial (cada centro comercial tiene un **número de departamentos**).

Se le brinda los siguientes segmentos de código como ayuda:

|  |  |
| --- | --- |
| **class** Obra{**protected:** **char** nombre[25]; **int** metros2; **float** precio;**public:** Obra(**char\***=’’sin nombre’’,**int**=0, **float**=0.0 ); **float** precioObra()**const**{**return** metros2\*precio;} **virtual int** tiempoObra() **const**=0; **char\*** getNombre() **const** {**return** nombre;} **int** getMetros2() **const** {**return** metros2;} **float** getPrecio() **const**{**return** precio;} **virtual** ~Obra(){}}; | **class** Hotel: **public** Obra{**protected:** **int** pisos;**public:** Hotel(**int** =0,**char**\*=”sin nombre”,**int**=0,**float**=0.0); **virtual int** tiempoObra()**const**; **int** getPisos()**const**{**return** pisos;}};**int** Hotel:: tiempoObra()**const**{ **return** 2\*pisos;}… |

Pruebe su implementación en Qt que realice lo siguiente:

1. Defina en el programa principal un arreglo de punteros a Obras si se sabe que la constructora manejará como máximo 5 obras. Solicite al usuario la adición de nuevas obras de cualquiera de los tres tipos al arreglo (creadas dinámicamente). Imprima en pantalla el tiempo de fabricación y el precio de cada obra empleando polimorfismo.
2. El programa debe permitir calcular el precio de la construcción completa.
3. El programa debe determinar el tiempo que demorará la construcción.
4. **Modifique el programa anterior para que en lugar de hacer el arreglo en el main(), haga una clase ArregloObras que encapsule el funcionamiento del arreglo que antes hacía en el programa principal, y use un objeto de dicha clase en el programa principal.**

**IMPLEMENTACIONES**

**Pregunta #1:**

[Texto del código]

[Imagen o imágenes que validen su correcta ejecución]

**Modificación pregunta #1:**

[Texto del código]

[Imagen o imágenes que validen su correcta ejecución]

**PREGUNTAS**

**¿Para qué se utilizan los métodos virtuales?**

**¿Qué implicaciones tiene un método virtual puro para el propio método y para la clase a la que pertenece?**

**¿Por qué se pone el destructor virtual en la clase base?**

**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**