

Actividad 3 - Lenguajes de programación II

Ingeniería en Desarrollo de Software

Tutor: Miguel Ángel Rodríguez

Alumno: José Luis Rodríguez Blancas

Fecha: 30/11/2023

Índice

Introducción	3
Descripción	3
Justificación	4
Desarrollo	5
Conclusión	7

Introducción

En el contexto de esta actividad, se busca desarrollar una sólida estructura de clases que permita a la empresa UNI gestionar de manera eficiente la información relativa a sus empleados. Para lograr esto, se emplearán conceptos fundamentales de programación orientada a objetos, como clases, herencia de clases y atributos. La aplicación resultante será capaz de manejar datos esenciales de los empleados, con un énfasis especial en la distinción entre empleados normales y directivos.

Los datos a gestionar incluyen elementos clave como el Número de Empleado (autogenerado y numérico), Nombre, Apellido Paterno, Apellido Materno, Fecha de Nacimiento, RFC (calculado en base al nombre y fecha de nacimiento), Centro de Trabajo, Puesto, Descripción del Puesto y una bandera para indicar si el empleado es un Directivo (1 para directivos, 0 para empleados normales).

Además, se debe tener en cuenta que los empleados de tipo Directivo presentan atributos adicionales, como el Número del Centro que supervisan (numérico y capturable) y una bandera que indica si reciben prestación de combustible. Este enfoque modular permitirá una adaptabilidad eficiente a las necesidades específicas de la empresa UNI, asegurando una gestión integral y organizada de la información de su personal.

Descripción

En este escenario, la empresa UNI busca establecer un sistema organizado y eficiente para gestionar la información de sus empleados. Para lograrlo, se plantea la necesidad de desarrollar una estructura de clases mediante programación orientada a objetos. Este enfoque implica la creación de clases que representen a

diferentes tipos de empleados, con la posibilidad de heredar atributos comunes y extender funcionalidades específicas.

La información a gestionar abarca aspectos cruciales de los empleados, como el Número de Empleado, Nombre, Apellido Paterno, Apellido Materno, Fecha de Nacimiento, RFC calculado a partir del nombre y fecha de nacimiento, Centro de Trabajo, Puesto y Descripción del Puesto. Además, se introduce una distinción entre dos tipos de empleados: los normales y los directivos. Esta diferenciación se logra mediante una bandera, donde el valor 1 indica un empleado directivo y 0 a un empleado normal.

Es esencial destacar que los empleados de tipo directivo poseen atributos adicionales, como el Número del Centro que supervisan, de naturaleza numérica y capturable, así como una bandera que indica si reciben prestación de combustible. Esta estructura modular y jerárquica permitirá a la empresa UNI adaptarse a sus necesidades específicas, facilitando la gestión integral de la información de su personal de manera coherente y eficaz.

Justificación

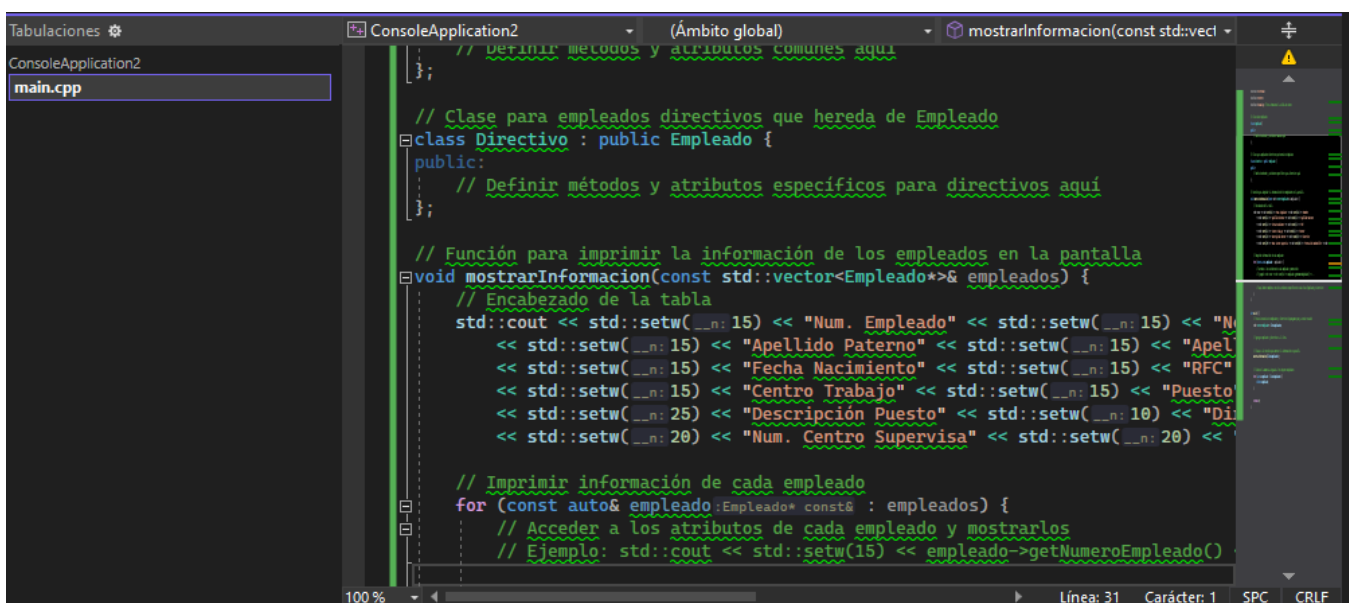
La adopción de una solución basada en clases, herencia de clases y atributos para gestionar la información de los empleados de la empresa UNI se justifica por varias razones fundamentales. En primer lugar, este enfoque ofrece una estructura organizada y modular que refleja fielmente la jerarquía y relaciones entre los diferentes tipos de empleados. La utilización de clases permite encapsular datos relacionados y comportamientos específicos, facilitando la comprensión y mantenimiento del código.

La herencia de clases proporciona una manera eficiente de gestionar la similitud entre empleados normales y directivos, permitiendo la reutilización de atributos comunes y la incorporación de características específicas para cada tipo. Esto promueve la coherencia en la representación de los datos y simplifica la extensión futura del sistema para incluir posibles variaciones en los tipos de empleados.

Además, el uso de atributos capturables y calculados, como el RFC, agiliza la entrada y procesamiento de datos, garantizando la integridad y precisión de la información. La inclusión de banderas para identificar el tipo de empleado proporciona una forma clara y eficaz de diferenciar entre empleados normales y directivos, simplificando las operaciones de gestión y consulta.

En resumen, la implementación de esta solución basada en clases y herencia proporciona una estructura robusta y flexible que se alinea con las necesidades específicas de la empresa UNI, facilitando la gestión integral y eficiente de la información de su personal.

Desarrollo



```

// Definir métodos y atributos comunes aquí
};

// Clase para empleados directivos que hereda de Empleado
class Directivo : public Empleado {
public:
    // Definir métodos y atributos específicos para directivos aquí
};

// Función para imprimir la información de los empleados en la pantalla
void mostrarInformacion(const std::vector<Empleado*>& empleados) {
    // Encabezado de la tabla
    std::cout << std::setw(15) << "Num. Empleado" << std::setw(15) << "N"
    << std::setw(15) << "Apellido Paterno" << std::setw(15) << "Apel"
    << std::setw(15) << "Fecha Nacimiento" << std::setw(15) << "RFC"
    << std::setw(15) << "Centro Trabajo" << std::setw(15) << "Puesto"
    << std::setw(25) << "Descripción Puesto" << std::setw(10) << "Di"
    << std::setw(20) << "Num. Centro Supervisa" << std::setw(20) << "

    // Imprimir información de cada empleado
    for (const auto& empleado : empleados) {
        // Acceder a los atributos de cada empleado y mostrarlos
        // Ejemplo: std::cout << std::setw(15) << empleado->getNumeroEmpleado()
    }
}

```

Este programa en C++ tiene como objetivo mostrar en pantalla la información de los empleados, considerando la estructura de clases y herencia definida en el contexto proporcionado.

1. ****Definición de Clases:****

- Se define una clase base llamada `Empleado` que contiene los atributos y métodos comunes para todos los empleados.

- Se crea una clase derivada llamada `Directivo` que hereda de la clase `Empleado` y agrega atributos y métodos específicos para los directivos.

2. ****Función `mostrarInformacion`:****

- Esta función toma un vector de punteros a objetos `Empleado` como parámetro.
- Utiliza la biblioteca `*iomanip*` para formatear la salida y presentar la información en una tabla.
- Imprime en la consola un encabezado de tabla con los nombres de los atributos.
- Itera a través del vector de empleados y, para cada empleado, accede a sus atributos utilizando los métodos correspondientes (que deberían estar implementados en las clases) y los muestra en la pantalla.

3. ****Función `main`:****

- En la función principal, se crea un vector `listaEmpleados` que contendrá punteros a objetos de la clase `Empleado`.
- Se supone que previamente has creado instancias válidas de empleados y directivos y las has agregado a la lista.
- Se llama a la función `mostrarInformacion` pasando la lista de empleados como argumento.
- Se libera la memoria asignada a los objetos de empleados al final del programa.

Conclusión

La implementación de una estructura de clases, herencia y atributos en el contexto de la actividad propuesta no solo representa un ejercicio práctico en el ámbito de la programación, sino que tiene implicaciones significativas en el campo laboral y la vida cotidiana. Al desarrollar un sistema que permite gestionar la información de los empleados de una empresa mediante estas técnicas de programación orientada a objetos, se establece una base sólida para la creación, organización y mantenimiento eficiente de datos.

En el ámbito laboral, la aplicación de estas prácticas brinda una herramienta poderosa para empresas como UNI al facilitar la adaptación a cambios en la estructura organizativa, la incorporación de nuevos tipos de empleados y la mejora continua del sistema. La modularidad y reutilización de código proporcionadas por la herencia de clases permiten una gestión ágil y escalable de la información, optimizando los procesos de seguimiento de empleados y simplificando la toma de decisiones basada en datos.

En la vida cotidiana, este enfoque también tiene implicaciones relevantes, ya que la programación orientada a objetos se encuentra en la base de numerosas aplicaciones y sistemas que utilizamos diariamente, desde aplicaciones móviles hasta plataformas web. Comprender y aplicar estos conceptos no solo mejora las habilidades de programación, sino que también potencia la capacidad para diseñar soluciones estructuradas y eficientes en diversos contextos, contribuyendo así al desarrollo profesional y al pensamiento lógico en general. En resumen, la importancia de esta actividad radica en su capacidad para proporcionar herramientas y habilidades esenciales que tienen un impacto positivo tanto en el entorno laboral como en la vida diaria.

Link

https://drive.google.com/file/d/1fZFX5KkZF1DBZGLkN_B1Y0H0BxVgriP/view?usp=sharing