**DESCRIPCIÓN DEL PROYECTO #1**

**“CINE 2.0”**

Objetivo

Desarrollar una venta de tickets para un cine en C++ utilizando para ello arreglos y matrices.

Descripción

NUEVA CINEMA SA es una nueva empresa dedicada al entretenimiento, para ello quiere iniciar su proyecto de un nuevo cine en el cantón de Pérez Zeledón, ya realizaron un estudio de mercado y se dieron cuenta que a pesar de contar con una franquicia de cines, es necesario abastecer en la demanda de entretenimiento en el cantón. Por ello, han determinado realizar un concurso y escoger la mejor solución informática que se presente en la fecha establecida. A continuación se describirá lo que esta empresa desea para su nuevo sistema de venta de tiquetes.

Especificaciones

1. **Menú:** El sistema deberá contener el siguiente menú de opciones:
   1. Archivo
      1. Acerca de
      2. Salir
   2. Mantenimiento
      1. Películas
      2. Salas
      3. Horarios
   3. Reserva
      1. Películas
   4. Venta
      1. Películas
2. **Archivo:** sólo se tendrán dos opciones: Acerca de, que muestra el nombre del estudiante que hizo el sistema (ser creativos). Y la opción de Salir del sistema.
3. **Mantenimiento:** se debe guardar la información de las películas, y también en otra opción guardar la información de las salas donde se verán dichas películas. Así como los horarios y precios. La información de la película a almacenar: nombre, año, duración (min), país, reseña. La información de la sala: número, cantidad butacas, precio. Para la información de los horarios: Día (fecha), hora inicial, hora final.
4. **Reserva:** en esta opción,primero se debe mostrar la película en la que se desea reservar y luego se debe mostrar la sala en la cual se exhibe. Luego el horario. Una vez elegida dicha selección, se mostrará la disponibilidad de las butacas para finalmente calcular el precio. Al finalizar este proceso, se generará un consecutivo. El cual será utilizado para la realización de la Venta.
5. **Venta:** en esta opción se debe incluir el consecutivo generado en el proceso anterior, es decir en la **Reserva**. Dicho otra forma, no se venderán tiquetes sin una reserva previa. Al momento de incluir el número de tiquete generado, el sistema procederá a pedir los datos del cliente (cédula y tarjeta). Y se desplegará el mensaje indicando que efectivamente se hizo el pago correspondiente.
6. Para la opción de **Reserva**, deben aparecer las butacas de las salas (matriz) con los identificadores, de tal forma que el usuario pueda entender cuál butaca está disponible, cuál está reservada y cuál está vendida. (Ser creativos).Se debe aclarar que los tiquetes reservados se pueden vender hasta que falten 30 minutos para la proyección de la película de lo contrario no pueden venderse.

Instrucciones Generales

1. El trabajo deberá entregarse en forma digital (fuentes y ejecutable) el día y hora indicados en el aula virtual según se indica en el cronograma del curso. El proyecto será desarrollado de **forma individual**.
2. Se evaluarán los siguientes puntos:
   1. Software (50 puntos)
      1. Funcionamiento (45 puntos)
      2. Defensa (5 puntos)
3. El proyecto será desarrollado en la herramienta ***VS Community 2022 - C++.***
4. El proyecto deberá llevar su control con una herramienta de control de versiones elegida por el profesor a cargo del curso. Para ello se hará una revisión de avance según las fechas establecidas.
5. El proyecto será probado en dos ocasiones, la primera será con el profesor y los estudiantes y la segunda será por parte del profesor a cargo del grupo.

**Fecha de Entrega**

Entrega del enunciado: Miércoles 30 de julio del 2024.

Revisión Avance: Miércoles 28 de agosto del 2024.

Revisión y defensa final: Miércoles 11 de setiembre del 2024.

Evaluación

El proyecto tendrá la siguiente evaluación, considerando aspectos como cumplimiento de funcionalidad y limpieza-claridad en el código, el cual debe ser orientado al uso de vectores y matrices:

|  |  |  |
| --- | --- | --- |
| **Aspectos a Evaluar** | | **Puntaje (50%)** |
| **Defensa.**  Se realizará una prueba al estudiante para determinar el grado de asimilación y entendimiento del desarrollo del proyecto entregado. | | **5%** |
| **Funcionamiento.**  El proyecto tendrá la siguiente evaluación a nivel de funcionamiento: | | **45%** |
|  | **Películas:** registro completo de la información de las películas. Utilizando vectores. | **5%** |
|  | **Salas:** registro completo de la información de las salas. Utilizando vectores. | **5%** |
|  | **Horarios:** registro completo de la información de los horarios. Utilizando vectores. | **5%** |
|  | **Reserva:** registro de la reserva. Para el manejo de la reserva y venta, será necesario el manejo de matrices para la escogencia de las butacas. | **20%** |
|  | **Venta:** desarrollo de las reglas del juego, tal y como se indicaron en el enunciado. | **10%** |
|  | **Faltas a las buenas prácticas:**  Menos de 5 faltas (Se le disminuye un 3%)  Menos de 6 - 8 faltas (Se le disminuye un 7%)  Más de 9 faltas (Se le disminuye un 10%)  Ver Lista de buenas prácticas en los anexos del programa de curso. | **-10% a 0%** |
|  | **TOTAL** | **50%** |

## Anexos

Anexo #1

**Buenas Prácticas solicitadas en el proyecto**

|  |  |
| --- | --- |
|  | **Código** |
|  | Utiliza estándares propios del lenguaje C++. |
|  | Utiliza control de versiones Git |
|  | Utiliza múltiples commits para la realización del proyecto, se denota una planeación en la realización del proyecto (Existe una cantidad de commits y orden razonable) |
|  | Cada commit corresponde a una tarea única, está bien comentado en inglés. (buena documentación) |
|  | Utiliza código correctamente formateado, escrito en inglés. |
|  | El código de cada archivo esta ordenado, variables declaradas primero, luego métodos públicos (empieza con el constructor y destructor) y de ultimo métodos privados |
|  | **Archivos** |
|  | Maneja archivos separados para cada estructura o clase, así como de encabezado. |
|  | El nombre de cada archivo cumple con las buenas prácticas: extensiones correctas, nombre representativo, utiliza los caracteres correctos. |
|  | **Nomenclatura** |
|  | El nombre de cada clase cumple con las buenas prácticas: nombre representativo, PascalCase, utiliza los caracteres correctos, es un sustantivo, está en inglés. |
|  | El nombre de cada atributo, variable o parámetro cumple con las buenas prácticas: nombre representativo, no incluye tipo, pero representa su identidad (por ejemplo, booleanos positivos, o nombres plurales), camelCase, utiliza los caracteres correctos, es un sustantivo, está en ingles |
|  | El nombre de cada método o función cumple con las buenas prácticas: nombre representativo, camelCase, utiliza los caracteres correctos, inicia con un verbo, indica correctamente lo que devuelve, realiza función única, muestra de manera explícita lo que debe recibir, está en ingles |
|  | **Estructuras de control** |
|  | Utiliza curly brackets siempre |
|  | Utiliza los diferentes ciclos para lo que es |
|  | Evita anidaciones de más de 3 en cada función |
|  | La lógica booleana esta optimizada, prefiere expresiones de condición legibles, utiliza el else solo cuando es necesario, utiliza mejores alternativas al switch |
|  | **Mensajes al usuario y comentarios** |
|  | El código no contiene comentarios |
|  | Los mensajes al usuario son positivos, están en español, tienen buena gramática y ortografía |
|  | **Programación orientada a objetos** |
|  | Utiliza herencia cuando es necesario |
|  | Utiliza polimorfismo cuando es necesario |
|  | Utiliza un nivel de abstracción adecuado |
|  | Utiliza encapsulamiento (abierto a extensión, cerrado para modificación) |
|  | Utiliza bajo acoplamiento y alta cohesión |
|  | Utiliza el principio de responsabilidad única |
|  | Utiliza el principio de DRY, reutiliza código, evita repetirlo |