**Justificación**

Las listas se utilizaron en este código debido a su flexibilidad, permitiendo manejar colecciones de elementos cuyo tamaño puede variar durante la ejecución del programa. Proporcionan métodos integrados para agregar, eliminar y verificar elementos fácilmente, lo que simplifica la manipulación de datos como las materias y matrículas de estudiantes. Además, las listas permiten un acceso rápido y eficiente a los elementos mediante índices y soportan iteración directa, facilitando operaciones como mostrar y procesar todos los elementos de una colección. Su compatibilidad con C# asegura que el código sea mantenible y entendible.

**Flexibilidad:**

Motivo: Las listas son una estructura de datos dinámica, lo que significa que su tamaño puede cambiar durante la ejecución del programa. Esto es especialmente útil en escenarios donde no se conoce de antemano el número exacto de elementos que se manejarán.

Aplicación en el Código: En el contexto de la matrícula universitaria, el número de materias que un estudiante puede seleccionar no está predeterminado y puede variar ampliamente. Las listas permiten agregar y eliminar materias según las necesidades de cada estudiante sin requerir una nueva estructura de datos o una reasignación de memoria.

**Facilidad de Uso:**

Motivo: Las listas en C# proporcionan una variedad de métodos integrados para manipular los elementos de la colección. Métodos como Add (para agregar elementos), Remove (para eliminar elementos), Contains (para verificar la existencia de un elemento) y Count (para obtener el número de elementos) simplifican enormemente la gestión de colecciones.

Aplicación en el Código: La facilidad de agregar y eliminar materias de la lista de materias seleccionadas por un estudiante mejora la eficiencia y la legibilidad del código. Del mismo modo, la capacidad de contar el número de materias seleccionadas permite calcular costos totales de manera sencilla.

**Acceso Rápido y Eficiente**:

Motivo: Las listas permiten acceder a sus elementos mediante índices, lo que facilita operaciones rápidas de lectura y escritura. El acceso indexado es particularmente útil cuando se necesita acceder frecuentemente a elementos específicos dentro de la colección.

Aplicación en el Código: En el programa de matrícula, es crucial poder acceder rápidamente a las materias disponibles para mostrarlas al usuario y permitir su selección. El acceso eficiente a los elementos asegura que el programa funcione de manera rápida y sin demoras perceptibles.

**Consistencia y Estabilidad**:

Aplicación en el Código: La elección de listas garantiza que el código sea mantenible y entendible para otros desarrolladores que puedan trabajar en el proyecto en el futuro, aprovechando una estructura de datos familiar y bien documentada.

**Biografía útil que nos ayudó a comprender mejor el uso de listas**

<https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/tutorials/arrays-and-collections>