UNIVERSIDAD NACIONAL, SEDE REGIONAL BRUNCA
EIF200 Fundamentos de Programación
Profesores: Jeanneth Alvarado Abarca, Rubén Mora Vargas, Juan Gamboa Abarca

PROYECTO #1
"conversor de sistemas numéricos"

Objetivo

Desarrollar una aplicación en C++ que permita la conversión de números entre sistemas numéricos (decimal, binario, octal y hexadecimal) mediante una interfaz interactiva y modular. La aplicación validará las entradas, gestionará errores y permitirá reutilizar el resultado de conversiones previas para facilitar operaciones sucesivas sin necesidad de reiniciar el programa.

Aspectos generales

1. Lea y comprenda cuidadosamente lo que se le solicite.
2. El Proyecto debe ser realizado individualmente
3. El día de entrega, el proyecto será defendido de manera individual. Se realizarán pruebas adicionales para conocer la autoría de su código y proyecto.
4. La nota será mérito personal por lo que deben conocer todas las particularidades del sistema y tener pruebas de trabajo realizado.
5. Si se comprueba que existen dos o más proyectos similares o copiados, se procederá a colocar nota cero a todos los proyectos involucrados.
6. Se requiere que suba su código al Aula Virtual en tiempo y forma.
7. Fechas importantes:
   a) Entrega del enunciado: Miércoles 5 de marzo del 2025
   b) Defensa Del Proyecto: Miércoles 9 de abril del 2025

Descripción del proyecto

Desarrolla una aplicación en **C++** que permita convertir números entre los siguientes sistemas numéricos:

- Decimal
- Binario
- Octal
- Hexadecimal

La aplicación contará con un menú interactivo que permita al usuario:

1. **Seleccionar el sistema de origen:**
   Elegir entre los sistemas disponibles (además de la opción de reutilizar el resultado de una conversión previa).
2. **Ingresar el número:**
   El usuario deberá ingresar un número en el sistema seleccionado. Se validará que el número contenga únicamente los dígitos permitidos para ese sistema (por ejemplo, en binario sólo se aceptarán 0 y 1). Se contemplarán números positivos y negativos.
3. **Seleccionar el sistema de destino:**
   El usuario elegirá a qué sistema desea convertir el número. Se implementará una validación para evitar que se seleccione el mismo sistema de origen y destino; en caso de hacerlo, se mostrará una advertencia solicitando una nueva opción.
4. **Mostrar el resultado:**
   El programa mostrará el resultado de la conversión de forma clara y organizada.

Requisitos y Funcionalidades

Modularización:

- Implementar funciones específicas para cada tipo de conversión.
- Estructurar el código de forma que se facilite la reutilización del resultado de la conversión previa.

Validación de Entrada:

- Verificar que el número ingresado sea válido para el sistema seleccionado (por ejemplo, en binario sólo se permitirán los dígitos 0 y 1).
- Permitir el ingreso de números tanto positivos como negativos.
- El programador definirá y propondrá los métodos de validación.

Interfaz Interactiva y Menú:

- Ofrecer un menú que permita seleccionar el sistema de origen, ingresar el número y elegir el sistema de destino.
- Incluir la opción de reutilizar el resultado de la conversión anterior, mostrando su sistema y valor.
- Evitar que el usuario seleccione el mismo sistema para origen y destino. En caso de hacerlo, se mostrará una advertencia y se solicitará una nueva opción.
- Permitir realizar múltiples conversiones sin necesidad de reiniciar la aplicación y ofrecer una opción clara para salir o reiniciar el proceso.

Conversión de Sistemas Numéricos:

- Realizar la conversión entre cualquier par de sistemas numéricos. No se requiere implementar conversiones directas sin pasos intermedios; se podrán usar pasos intermedios (por ejemplo, pasar por el sistema decimal) si es necesario.
- El programador podrá definir los métodos o fórmulas de conversión según se considere apropiado.

Manejo de Errores y Retroalimentación:

- Mostrar mensajes claros de error o advertencia cuando el usuario ingrese datos inválidos o seleccione opciones incorrectas.
- Informar de manera adecuada al usuario en cada paso del proceso.

Memoria de Conversión:

- Almacenar el último resultado de conversión y el sistema de destino utilizado.
- Permitir que, en una conversión posterior, el usuario pueda reutilizar ese resultado como nuevo valor de origen sin tener que ingresarlo nuevamente.

Ejemplo de uso

--- Conversor de Sistemas Numéricos ---

Seleccione el sistema de origen:

1. Decimal

2. Binario

3. Octal

4. Hexadecimal

Opción: 2

Ingrese el número en sistema binario: 101101

Seleccione el sistema al que desea convertir:

1. Decimal

2. Octal

3. Hexadecimal

Opción: 3

El número 101101 en hexadecimal es: 2D

¿Desea realizar otra conversión? (S/N): S

Seleccione el sistema de origen:

0. Usar resultado previo (Sistema: Hexadecimal, Valor: 2D)

1. Decimal

2. Binario

3. Octal

4. Hexadecimal

Opción: 0

Número previo detectado: 2D

Seleccione el sistema al que desea convertir:

1. Decimal

2. Binario

3. Octal

Opción: 1

El número 2D en decimal es: 45

¿Desea realizar otra conversión? (S/N): N

Criterios de evaluación

Como directriz de la cátedra se establece como obligatoria la defensa del proyecto, por parte del estudiante, la nota será siempre tomada de manera individual.

La presente rúbrica desglosa la evaluación del proyecto según en los siguientes rubros:

Rúbricas

La nota del trabajo se calcula en escala de 0 a 100, según las siguientes rúbricas:

- Eficacia: 70%. Cumple con los puntos solicitados

Rúbrica	100%	80% - 60%	60% - 40%	40% - 0%
Menú interactivo	Menú completo con todas las opciones funcionales, incluyendo reutilizar resultado previo.	Falta alguna opción del menú o tiene errores menores en su funcionamiento.	Menú incompleto o con fallos graves en la navegación.	No hay menú interactivo o es inutilizable.
Validación de entrada	Valida correctamente dígitos, signos y sistemas numéricos en todos los casos.	Valida la mayoría de casos, pero falla con números negativos o caracteres especiales.	Validación parcial (ej: solo verifica algunos sistemas).	No hay validación de entrada o es incorrecta.
Conversiones precisas	Todas las conversiones entre sistemas son correctas en cualquier combinación.	Conversiones correctas, pero con errores en casos específicos (ej: números grandes).	Solo funciona para algunas combinaciones de sistemas.	Las conversiones no son correctas o no se implementan.
Manejo de errores	Detecta y muestra mensajes claros para entradas inválidas y sistemas iguales.	Maneja errores básicos, pero sin mensajes claros o falta algunas validaciones.	Solo detecta algunos errores o los mensajes son confusos.	No hay manejo de errores o es completamente inefectivo.
Memoria de conversión	Almacena y permite reutilizar el último resultado correctamente.	Permite reutilizar el resultado, pero no almacena el sistema destino.	Funcionalidad implementada parcialmente con errores frecuentes.	No hay memoria de conversión o no funciona.
Modularización	Código bien estructurado con funciones específicas para cada conversión.	Modularización básica, pero con lógica repetida o funciones mezcladas.	Estructura deficiente con código monolítico y poca reutilización.	No hay modularización (código espagueti).
Interfaz y flujo	Muestra resultados claros, permite múltiples conversiones y cierre controlado.	Flujo funcional, pero con formato desorganizado o reinicios forzados.	Interfaz confusa o requiere reiniciar el programa cada vez.	No muestra resultados o el flujo es caótico/inutilizable.

- Eficiencia: 15%.

Consideración General	100%	80% - 60%	60% - 40%	40% - 0%
Claridad y Estructura del Código	El código está altamente estructurado, modularizado y fácil de entender, con comentarios claros que explican la lógica detrás de cada sección. Se siguen buenas prácticas de programación y se evitan redundancias.	El código es en su mayoría claro y estructurado, pero puede haber algunas secciones difíciles de entender o falta de comentarios en partes críticas. Se siguen algunas buenas prácticas de programación.	El código es difícil de entender y carece de estructura, con pocas o ninguna explicación sobre su funcionamiento. Se observan redundancias y falta de coherencia en la implementación.	El código es confuso, desordenado y difícil de seguir, con múltiples errores de sintaxis y lógica. No se siguen prácticas de programación adecuadas.
Eficiencia del Algoritmo y Tiempo de Ejecución	El proyecto demuestra un alto nivel de eficiencia en el diseño de algoritmos, con tiempos de ejecución rápidos y un consumo de recursos óptimo. Se implementan algoritmos adecuados para las tareas requeridas.	El proyecto es en su mayoría eficiente en términos de algoritmos y tiempo de ejecución, pero puede haber algunas áreas donde se podrían realizar mejoras para optimizar el rendimiento. Algunos algoritmos pueden	El proyecto tiene problemas de eficiencia, con tiempos de ejecución lentos o un consumo de recursos excesivo en ciertas partes del código. Se pueden identificar áreas donde se pueden hacer mejoras significativas en términos de rendimiento.	El proyecto es altamente ineficiente en términos de algoritmos y tiempo de ejecución, con tiempos de ejecución extremadamente lentos o un consumo de recursos excesivo en la mayoría de las operaciones. Es necesario revisar y

Consideración General	100%	80% - 60%	60% - 40%	40% - 0%
		no ser los más adecuados para ciertas tareas.		reestructurar completamente el código para mejorar su eficiencia.
Mantenibilidad del Código	El código es fácilmente mantenible y extensible, con una arquitectura bien definida que permite realizar cambios y agregar nuevas funcionalidades de manera eficiente y sin afectar la funcionalidad existente.	El código es en su mayoría mantenible, pero puede haber algunas áreas que requieran mejoras en términos de modularidad y extensibilidad.	El código presenta dificultades para el mantenimiento y la extensión debido a una estructura poco clara o a la falta de abstracción y reutilización de código.	El código es altamente propenso a errores y difícil de mantener o extender debido a una estructura caótica y una falta de modularidad y abstracción.
Manejo de Errores y Excepciones	El programa maneja de manera efectiva todos los posibles errores y excepciones que pueden surgir durante la ejecución, proporcionando mensajes claros al usuario y evitando fallos inesperados.	El programa maneja la mayoría de los errores y excepciones de manera adecuada, pero puede haber algunos casos donde la gestión de errores no sea óptima o los mensajes de error no sean lo suficientemente informativos.	El programa tiene problemas en el manejo de errores y excepciones, con fallos inesperados o mensajes de error confusos que dificultan la comprensión del usuario.	El programa no maneja adecuadamente los errores y excepciones, lo que resulta en fallos frecuentes y mensajes de error poco claros o inexistentes.

- Buenas prácticas: 15%.

Consideración General	100%	80% - 60%	60% - 40%	40% - 0%
Cantidad de aplicación a las buenas practicas	El código esta bien refactorizado, tiene menos de 3 faltas	El código esta bastante refactorizado, tiene menos de 10 faltas	El código tiene menos de 15 faltas, está un poco descuidado o no presenta buen esmero en aplicar las buenas prácticas.	El código tiene mas de 15 faltas, este descuidado o presenta nulo esmero en aplicar las buenas prácticas.