

Semana de Integración I – Matemática – Programación I

Trabajo Integrador

“Contador Binario”

Alumnos – Comisión Nº 23, Grupo Y

Guido Damian Agüero, Juan Cruz Alemis, Sol Yoon, Luciano Wittmund y Leandro Zalazar

Matemática

Docente Titular : Martina Wallace

Docente Tutor: Sebastián Marinier

Programación I

Docente Titular : Nicolás Quirós

Docente Tutor: Giuliano Crenna

26 de Abril de 2025

Contador Binario

Descripción del Proyecto:

El proyecto consiste en un programa diseñado para convertir números decimales en números binarios de 4 dígitos.

Entradas: El programa utiliza un ciclo que cuenta en este caso desde 0 hasta 15

Resolución del problema: El programa utiliza dos funciones principales y una extensión:

concatenar_ceros(rs) que rellena la cadena binaria con ceros al inicio para asegurar que siempre tenga 4 dígitos.

conversor_binario(i) que realiza la conversión del número decimal al binario mediante cálculos con el operador módulo % y la división entera //. En el caso especial para el número 0, asigna directamente el formato binario 0000.

El código también incluye una pausa de 0.5 segundos entre cada conversión que le da la extensión **time.sleep()** para mejorar lectura al momento que imprime por pantalla.

Salida: El programa genera como salida, una lista de números decimales junto con su representación binaria, utilizando un retardo de 0.5 segundos.

```
0: 0000
1: 0001
2: 0010
3: 0011
4: 0100
5: 0101
6: 0110
7: 0111
8: 1000
9: 1001
10: 1010
11: 1011
12: 1100
13: 1101
14: 1110
15: 1111
```

Video explicativo en Youtube:

https://www.youtube.com/watch?v=PwgzwwQA_h0

Código fuente en Python:

```
# Contador Binario:

# Escriban un programa que, usando un ciclo, cuente desde 0 hasta 15 y muestre
cada número en su representación binaria.

# Extensión: Utilicen un retardo (por ejemplo, con time.sleep) para simular el
conteo de un circuito.

import time

def concatenar_ceros(rs): # Función para rellenar cifras 0 al número binario
    while len(rs) < 4:
        rs = "0" + rs
    return rs

def conversor_binario(i): # Función que convierte decimal a binario
    if i == 0:
        print(f"{i}: 0000") # Maneja el caso especial para 0
    else:
        resto_string = "" # Variable para almacenar el binario
        numero = i
        while numero > 0: # Convierte el número a binario
            resto = numero % 2 # Obtiene el resto
```

```
        resto_string = str(resto) + resto_string # Agrega el dígito al
inicio
        numero = numero // 2 # Divide entre 2
        resto_string = concatenar_ceros(resto_string) # Asegura que tenga 4
dígitos
        print(f"{i}: {resto_string}")

for i in range(16):
    conversor_binario(i) #Llama a la función para convertir i a binario y
mostrarlo por pantalla
    time.sleep(0.5) # Pausa de 0.5 segundo
```

Uso de Inteligencia Artificial

Durante el desarrollo del trabajo práctico, utilizamos ChatGPT —un chatbot de inteligencia artificial desarrollado por OpenAI capaz de mantener conversaciones en lenguaje natural, responder preguntas y generar texto de manera coherente— como herramienta de apoyo. A partir de la consigna planteada, solicitamos ejemplos de código relacionados, los cuales nos sirvieron de referencia para estructurar, definir funciones y establecer la lógica del programa. De esta manera, integramos las propuestas sugeridas respetando las buenas prácticas en el uso de inteligencia artificial en el ámbito académico.

Necesito hacer una código en python con la siguiente consigna "- Integrar los conceptos matemáticos de las 2 unidades vistas hasta ahora (Álgebra de Boole, sistema binario)."

Perfecto. Para integrar **Álgebra de Boole** y **sistema binario** en un código Python sencillo, se puede hacer un programa que:

- Pida al usuario ingresar dos números binarios.
- Realice operaciones lógicas (AND, OR, NOT, XOR) sobre ellos.
- Muestre los resultados tanto en binario como en decimal.

Aquí tenés un ejemplo de código que cumple con esa consigna:

```
python Copy Edit

def binario_a_decimal(binario):
    return int(binario, 2)

def decimal_a_binario(decimal):
    return bin(decimal)[2:]

def aplicar_operaciones(bin1, bin2):
    dec1 = binario_a_decimal(bin1)
    dec2 = binario_a_decimal(bin2)

    and_result = dec1 & dec2
    or_result = dec1 | dec2
    xor_result = dec1 ^ dec2
    not1_result = ~dec1 & ((1 << len(bin1)) - 1) # NOT Limitado a la Longitud de
    not2_result = ~dec2 & ((1 << len(bin2)) - 1)

    print(f"\nOperaciones lógicas entre {bin1} y {bin2}:")
    print(f"AND : {decimal_a_binario(and_result)} (Decimal: {and_result})")
    print(f"OR : {decimal_a_binario(or_result)} (Decimal: {or_result})")
    print(f"XOR : {decimal_a_binario(xor_result)} (Decimal: {xor_result})")
    print(f"NOT {bin1}: {decimal_a_binario(not1_result)} (Decimal: {not1_result})")
    print(f"NOT {bin2}: {decimal_a_binario(not2_result)} (Decimal: {not2_result})")

# Programa principal
print("Operaciones con Álgebra de Boole y Sistema Binario")
binario1 = input("Ingrese el primer número binario: ")
binario2 = input("Ingrese el segundo número binario: ")

# Validación básica
if len(binario1) != len(binario2):
    print("Los números binarios deben tener la misma longitud.")
else:
    aplicar_operaciones(binario1, binario2)
```

¿Qué integra?

- **Sistema binario:** el usuario introduce números binarios y se convierten a decimal.
- **Álgebra de Boole:** se aplican operaciones booleanas (AND, OR, XOR, NOT) entre esos números.

¿Querés que este código incluya también una interfaz gráfica sencilla o que lo convierta en un script interactivo más avanzado?

Me gustaria que este mas simplificado, utilizando funciones

Perfecto, acá tenés una versión más **simplificada y funcional**, ideal para empezar. El código está organizado con funciones pequeñas y claras, manteniendo todo lo esencial:

```
python
def bin_a_dec(b):
    return int(b, 2)

def dec_a_bin(d, longitud):
    return format(d, f'0{longitud}b')

def operaciones_logicas(b1, b2):
    d1 = bin_a_dec(b1)
    d2 = bin_a_dec(b2)
    n = len(b1)

    print(f"\nAND : {dec_a_bin(d1 & d2, n)}")
    print(f"OR  : {dec_a_bin(d1 | d2, n)}")
    print(f"XOR : {dec_a_bin(d1 ^ d2, n)}")
    print(f"NOT {b1}: {dec_a_bin(~d1 & (2**n - 1), n)}")
    print(f"NOT {b2}: {dec_a_bin(~d2 & (2**n - 1), n)}")

def main():
    print("Operaciones con Álgebra de Boole y Binario")
    b1 = input("Primer binario: ")
    b2 = input("Segundo binario: ")

    if len(b1) != len(b2):
        print("Error: deben tener la misma cantidad de bits.")
    else:
        operaciones_logicas(b1, b2)

main()
```

Ventajas:

- Cada función hace una sola cosa.
- Fácil de entender para alguien que empieza.
- Claramente vincula los temas de la consigna.

¿Querés que le agregue comentarios o ejemplos de uso para practicar?

Me gustaria simplificarlo aun mas, utilizando estructuras de decision y de repeticion

¡Perfecto! Vamos a hacer una versión aún más simple que use estructuras de **decisión** (`if`) y **repetición** (`while`), y reduzca al mínimo el uso de funciones, manteniéndolo claro y educativo:

```
python Copy Edit

print("Operaciones con Álgebra de Boole y Sistema Binario")

while True:
    b1 = input("Ingrese el primer número binario: ")
    b2 = input("Ingrese el segundo número binario: ")

    if len(b1) != len(b2):
        print("Error: deben tener la misma cantidad de bits. Intente de nuevo.\n")
    else:
        break

# Convertir binario a decimal
d1 = int(b1, 2)
d2 = int(b2, 2)
n = len(b1)

# Operaciones Lógicas
and_result = d1 & d2
or_result = d1 | d2
xor_result = d1 ^ d2
not1_result = ~d1 & (2**n - 1)
not2_result = ~d2 & (2**n - 1)

# Mostrar resultados en binario
print("\nResultados:")
print("AND :", format(and_result, f'0{n}b'))
print("OR  :", format(or_result, f'0{n}b'))
print("XOR :", format(xor_result, f'0{n}b'))
print(f"NOT {b1} :", format(not1_result, f'0{n}b'))
print(f"NOT {b2} :", format(not2_result, f'0{n}b'))
```

¿Qué usa este código?

- `if` para validar los datos.
- `while` para repetir la entrada hasta que esté bien.
- Sin funciones externas para que todo esté a la vista.
- Integra Álgebra de Boole y sistema binario de forma clara.

¿Querés agregarle un menú para elegir qué operación hacer?

Conclusión

En conclusión, el proyecto consolidó habilidades de programación, destacó la importancia de las buenas prácticas y mostró cómo las herramientas tecnológicas pueden mejorar el proceso de desarrollo y permitió aplicar conceptos clave de programación, como el uso de ciclos y funciones, para convertir números decimales en binarios. El uso de ChatGPT como herramienta de apoyo contribuyó a una mejor organización de las ideas del grupo.